Ein Framework für die profilbasierte Gruppenbildung in ad hoc Umgebungen

Dissertation

zur Erlangung des Doktorgrades
im Fach Informatik

eingereicht an der
Fakultät für Angewandte Informatik
Universität Augsburg

von
Christian Seitz
geb. am 25.02.1975 in Eichstätt

Dekan: Prof. Dr. Wolfgang Reif

Gutachter: 1. Prof. Dr. Bernhard Bauer
2. Prof. Dr. Theo Ungerer

Eingereicht: 11.04.2005
Tag der Promotion: 20.05.2005

Danksagung

Die hier vorliegende Arbeit entstand während meiner Zeit als Doktorand im Fachzentrum für Intelligente Autonome Systeme der Siemens AG in München. In der Zeit, die ich dort verbrachte, haben viele Personen in der einen oder anderen Form zum Entstehen der Arbeit beigetragen. Ich möchte mich an dieser Stelle für ihre Unterstützung bedanken.

Mein Dank gilt Herrn Prof. Dr. Bernhard Bauer, der die Betreuung der Arbeit übernahm. Er bot mir von Anfang an die Möglichkeit, sehr selbständig zu arbeiten und ließ mir auch inhaltlich viele Freiheiten. Bei Fragen und Problemen stand er mir jedoch stets mit gutem Rat und wertvollen Vorschlägen zur Seite.
Bei Herrn Prof. Dr. Theo Ungerer bedanke ich mich für die Übernahme des Zweitgutachtens.

Mein besonderer Dank geht an Dr. Michael Berger. Er war maßgeblich an der Aufgabenstellung beteiligt und hat mich während der kompletten Arbeit richtungsweisend begleitet. Er unterstützte mich durch zahlreiche Diskussionen und hat durch wichtige Hinweise die Struktur der Arbeit beeinflusst.

Dr. Markus Jäger und Dr. Thomas Wösch danke ich für die akribische und kritische Durchsicht des Skriptes und die hiermit verbundenen Anregungen und Verbesserungen. Dr. Markus Jäger möchte ich noch zusätzlich für seine konstruktiven Hinweise und seine Diskussionsbereitschaft im Verlauf der Arbeit meinen Dank aussprechen.

Weiterhin möchte ich mich bei Stefanie Kieweg für das Korrekturlesen bedanken.

Besonders danken möchte ich meinen Kollegen Michael Watzke und Dmitri Toropov, mit denen ich über drei Jahre das Büro teilte. Während dieser Zeit führten wir unzählige Diskussionen - nicht nur fachlicher Natur - und tauschten Ideen aus. Beide sorgten überdies für eine anregende, humorvolle und angenehme Atmosphäre in Zimmer 33-542.

Zusätzlich möchte ich mich bei Herrn Rudolf Kober bedanken, der in seiner Eigenschaft als mein Vorgesetzter mir diese Arbeit erst ermöglichte.

Schließlich möchte ich all denen danken, die mir durch Rat und Tat die Arbeit erleichtert und den Rücken gestärkt haben.

Kurzfassung

Die großen Fortschritte im Bereich drahtloser Kommunikationssysteme und mobiler Endgeräte eröffnen neue Anwendungsgebiete. Mit kleinen Geräten wie Notebooks, PDAs und Mobiltelefonen werden zunehmend verteilte, mobile ad hoc Anwendungen entwickelt, die dem Benutzer jederzeit zur Verfügung stehen.

Im Mittelpunkt dieser Arbeit steht die Formalisierung, Entwicklung und Evaluierung des Frameworks MoPiDiG - *Mobile Profile based Distributed Grouping*. Kernfunktionalität dieses Frameworks ist die dynamische kontextsensitive Klassifikation von Personen in virtuellen Gruppen in ad hoc Umgebungen. In MoPiDiG ist jeder Benutzer mit einem mobilen Endgerät und einem Benutzerprofil ausgestattet. Während sich die Benutzer bewegen, werden ähnliche Profile gefunden und die Benutzer zu Gruppen zusammengeschlossen. Die Teilnehmer einer Gruppe können miteinander kooperieren oder ein gemeinsames Ziel erreichen.

Im Vergleich zu bereits existierenden Verfahren verzichtet das MoPiDiG Framework auf zentrale Komponenten und ist somit komplett dezentral organisiert. Ferner arbeitet MoPiDiG ohne zusätzliche Ortsinformation. Zum Nachrichtenaustausch wird ausschließlich lokale Kommunikation, wie z. B. WLAN oder Bluetooth, verwendet.

Zur Realisierung von MoPiDiG ist die Lösung von vier separaten Kernproblemen notwendig. Auf Grund der Mobilität der MoPiDiG Benutzer kann die Kommunikation zwischen Teilnehmern abbrechen, was den Gruppierungsvorgang erheblich erschwert. Zusätzlich sind Methoden notwendig, die aus einer Menge an Profilen die essentielle Teilmenge zu Gruppen zusammenfasst. Um den Gruppierungsprozess in MoPiDiG zu beschleunigen, sind Optimierungsprozesse notwendig, die mit Hilfe von Heuristiken umgesetzt werden. Zusätzlich wird die Anzahl an Kommunikationspartnern gegebenenfalls mit Hilfe von Segmentierungsmethoden reduziert.

Zur Umsetzung von MoPiDiG wurde eine modulare Architektur entwickelt, die zum einem aus einer domänenunabhängigen Schicht besteht, welche aus Algorithmen und Meta-Modellen besteht. Zum anderen sorgt eine domänenabhängige Schicht dafür, dass MoPiDiG mit möglichst wenig Aufwand an neue Domänen angepasst werden kann.

Neben dem allgemeinen Aufbau von MoPiDiG wird zur Evaluierung ein Anwendungsbeispiel näher untersucht. Hierbei handelt es sich um dynamisches Taxi-Sharing. Ausgangspunkt für dieses Szenario sind stark frequentierte Orte, wie z. B. ein Bahnhof. Während die Benutzer sich noch im Zug befinden oder bereits im Bahnhofsgebäude aufhalten, wird die Profilinformation ausgetauscht und auf diese Weise versucht, eine oder mehrere Gruppen zu erzeugen. Zu einer Gruppe schließen sich Teilnehmer zusammen, wenn sich eine Nutzenfunktion erhöht. Dies ist der Fall, wenn Personen das gleiche Reiseziel haben, oder ein Weg existiert, der ihre Ziele ohne größere Umwege erreicht.

Zur Evaluierung des Frameworks wurde eine Simulationsumgebung für MoPiDiG Anwendung erzeugt. Diese Simulationsumgebung wurde verwendet um Aussagen über die Stabilitätszeit der erzeugten Gruppen, der Skalierbarkeit des Gruppierungsprozesses und die Effizienz der verwendeten Heuristiken machen zu können. Die Ergebnisse zeigen, dass durch Anwendung der Heuristiken die Ausführungszeit des Gruppierungsprozesses derart reduziert wird, dass eine Anwendung auf mobilen Endgeräten möglich ist.

Inhaltsverzeichnis

Kapitel 1

Einleitung

In den letzten Jahren haben immer mehr mobile Endgeräte (z. B. PDAs, Mobiltelefone) Einzug in den Alltag gehalten. Mit ihnen ist es bisher möglich, an jedem Ort auf Daten zuzugreifen und Anwendungen im Internet zu nutzen. Ein Beispiel hierfür ist eine Applikation für ein Mobiltelefon, mit welcher Routenplanerfunktionalitäten im Kfz erreicht werden.

Verteilte Anwendungen mit mehreren mobilen Endgeräten sind derzeit kaum vorhanden. Existieren sie doch, verfügen sie ausnahmslos über einen zentralen Koordinator und benötigen zusätzliche Infrastruktur.

In dieser Arbeit wird eine Framework für mobile Endgeräte vorgestellt, mit dem es möglich ist ohne zusätzliche Infrastruktur Personen auf Basis von Profildaten zu Gruppen zusammenzufassen und deshalb als *Mobile Profile based Distributed Grouping* (kurz: MoPiDiG) bezeichnet wird.

1.1 Profilbasierte Gruppenbildung in ad hoc Netzwerken

Im MoPiDiG Framework besitzt jeder Benutzer ein mobiles Endgerät, auf welchem sich sein Benutzerprofil befindet. Ein Benutzerprofil, im Weiteren oft kurz als Profil bezeichnet, ist eine umfassende Datensammlung zu einem bestimmten Objekt, in diesem Fall einer Person. Ein Benutzer kann in dieses Profil sowohl gewöhnliche Daten wie Name, Wohnort, etc. eintragen als auch Präferenzen ablegen.

Für eine MoPiDiG Anwendung wird keine zusätzliche Infrastruktur benötigt. Die Kommunikation zwischen den Endgeräten findet mit so genannten spontanen (ad hoc) Netzwerken statt, die sich dann bilden, wenn sich Geräte in Kommunikationsreichweite befinden. Innerhalb der Kommunikationspartner versucht MoPiDiG Gruppen zu finden. Eine Gruppe zeichnet sich dadurch aus, dass deren Mitglieder ähnliche Profile aufweisen. Was unter „ähnlich“ zu verstehen ist, bestimmt die jeweilige Anwendungsdomäne. Die Tatsache, dass sich mehrere Personen zu einer Gruppe formieren, muss einen Nutzen für die Gruppenmitglieder bringen.

Mit der Entwicklung des MoPiDiG Frameworks sind erhebliche Schwierigkeiten verbunden. Da bei jeder Anwendung des MoPiDiG Frameworks der Anwender zur Ausführungszeit mobil ist und sich frei bewegen kann, muss mit einer sehr schwierigen Kommunikationssituation gerechnet werden. Kommunikationsverbindungen stehen i. A. nur

eine gewisse Zeit zur Verfügung, bevor sie mobilitätsbedingt spontan beendet werden. Innerhalb dieser kurzen Zeitspanne muss der Profilaustausch mit sämtlichen potenziellen Gruppenmitgliedern abgeschlossen sein, da diese Daten für die Gruppenbildung benötigt werden.

Als eine zusätzliche Teilproblematik muss in MoPiDiG ein Mechanismus existieren, der die Anzahl der potentiellen Gruppierungspartner reduziert, wenn diese Menge zu groß ist. Lösbar ist dies dadurch, dass das zu Grunde liegende Netzwerk in der Größe beschränkt wird, indem es in Segmente aufgeteilt wird.

Des Weiteren müssen für das MoPiDiG Framework aus einer Datenmenge ähnliche Daten gefunden werden. Hier existieren bereits Methoden aus dem Data Mining bzw. dem Datenbankbereich. In diesen Gebieten liegen die Daten jedoch zentral vor und nicht verteilt, wie im Falle von MoPiDiG.

Tabelle 1.1 fasst alle zu lösenden Teilproblematiken in MoPiDiG zusammen und stellt

	mobile ad hoc Netzwerke	Segmentierung in Netzwerken	Clustering in Datenbanken	MoPiDiG
Kommunikation in mobilen Umgebungen	★			★
Aufteilen eines Kommunikationsnetzwerks in Teilnetze		★		★
Finden von ähnlichen Daten aus einer willkürlichen Datenmenge			★	★
Optimierungsverfahren				★

Tabelle 1.1: Benötigte Basistechnologien für das MoPiDiG Framework

erste existierende Lösungsmöglichkeiten gegenüber. Jedoch können diese Lösungen nicht in ihrer existierenden Form angewandt werden, sondern es bedarf noch Anpassungen. Zudem müssen die einzelnen Teilprobleme miteinander zu einer homogenen Lösung verbunden werden.

Zusätzlich sind in Tabelle 1.1 noch Optimierungsverfahren für MoPiDiG als notwendig erachtet. Dies folgt aus der Tatsache, dass das MoPiDiG Framework für mobile Endgeräte konzipiert sein soll. Mobile Geräte verfügen generell über beschränkte Ressourcen. Prozessorgeschwindigkeit und Speicherplatz stehen nur begrenzt zur Verfügung. Neben der Ausführungszeit ist bei mobilen Anwendungen auch Energieverbrauch ein Effizienzparameter, da Energie meist nur für eine gewisse zeitliche Periode zur Verfügung steht. Aus diesem Grund stellen Optimierungsverfahren einen wesentlichen Bestandteil des MoPiDiG Frameworks dar.

Wie bei allen mobilen Anwendungen sind bei MoPiDiG auch Sicherheitsaspekte, wie z. B. sicherer Profilaustausch zu berücksichtigen. Diese Fragestellung wird jedoch nur am Rande beantwortet, da ihre Lösung über den Rahmen der Arbeit hinausgehen würde.

Im MoPiDiG Framework wird im Gegensatz zu den meisten existierenden Gruppie-

rungsanwendungen zum Nachrichtenaustausch nur lokale Kommunikation wie Bluetooth und WLAN verwendet. Diese Technologien sind für die Gruppenbildung ausreichend und verursachen im Gegensatz zu GSM oder UMTS keine Kosten.

Die Wahl der ausschließlichen Verwendung lokaler Kommunikation hat Auswirkungen auf die Architektur. So darf im MoPiDiG Framework keine zentrale Komponente existieren, weil sämtliche Anwender mobil sind. Ein Koordinator könnte sich jeder Zeit von seinen zu koordinierenden Geräten entfernen und es müsste ein neuer Koordinator gewählt werden, was zu viel Zeit in Anspruch nehmen würde.

Weiterhin zeichnet das MoPiDiG Framework aus, dass MoPiDiG nicht auf eine spezielle Domäne begrenzt ist, d. h. es wird eine domänenunabhängige Architektur angestrebt. Eine völlig domänenunabhängige Lösung ist schwer zu realisieren. In MoPiDiG ist es jedoch mit wenig Zusatzaufwand möglich, verschiedene Domänen zu erschließen.

Ferner ist zur Gruppenbildung im Vergleich mit anderen Gruppierungsverfahren in MoPiDiG keine Ortsinformation oder Geo-Koordinate zwingend nötig, da diese nicht generell voraussetzbar ist, wie z. B. in Gebäuden. Sollten solche Daten jedoch zur Verfügung stehen, können und sollen sie auch genutzt werden, weil auf diese Weise der Gruppierungsprozess beschleunigt und verbessert werden kann.

1.1.1 Gruppenbildung

Mittelpunkt dieser Arbeit stellt die Gruppenbildung dar. Aus diesem Grund muss erörtert werden, warum dieser Prozess nötig ist und Anwendungen zur Gruppenbildung entwickelt werden sollen. Wird der Mensch als Beispiel für die Gruppenbildung herangezogen, so gibt es primär Gründe, warum sich *Personen* generell zu Gruppen zusammenschließen.

Eine Anzahl an Objekten (Menschen oder Maschinen) schließen sich zu einer Gruppe zusammen, da sie einzeln nicht in der Lage sind, ein gewisses Ziel zu erreichen. Die Gruppe ist somit Voraussetzung, um das Ziel zu erreichen. Soll beispielsweise ein komplexes Werkstück gefertigt werden, so werden hierfür mehrere Maschinen oder Spezialisten benötigt. Auch für viele Freizeitaktivitäten, vor allem Sportarten wie z. B. Fußball, ist eine Gruppe notwendig.

Im anderen Fall kann von jedem Objekt das Ziel auch allein erreicht werden. Durch Zusammenschluss zu einer Gruppe kann dieses jedoch schneller, besser oder mit weniger Aufwand erreicht werden. Wie sich dieser Synergieeffekt im Einzelnen bemerkbar macht, ist von der jeweiligen Anwendung abhängig. Es können sich durch die Bildung einer Gruppe beispielsweise die Kosten für ein Produkt reduzieren, indem spezielle Gruppentarife ausgenutzt werden können.

1.1.2 Anwendungsgebiete

Das Framework ist auf kein Anwendungsgebiet beschränkt. Es kann dort Anwendung finden, wo sich mehrere potenzielle Nutzer auf einem lokal eingeschränkten Raum befinden. Dies können Marktplätze, Biergärten, Bahnhöfe, Flughäfen oder sonstige stark frequentierte Einrichtungen sein. Ein Benutzer des Systems muss nicht „menschlich“ sein, auch Maschinen kommen in Betracht, sofern sie mit einem Profil ausgestattet werden können. Zusätzlich sollen noch folgende Anforderungen erfüllt sein:

- Die Bildung einer Gruppe muss motiviert sein, d. h. die Gruppe muss sich durch einen Synergieeffekt oder deren Notwendigkeit zur Problemlösung auszeichnen, was durch eine Nutzenfunktion angegeben werden kann.

- Die Gruppenbildung soll in einem dynamischen Umfeld erfolgen, d. h. sie ist dadurch gekennzeichnet, dass die Teilnehmer sich bewegen. Damit ist die Kommunikationsdauer eingeschränkt. Das Verfahren kann zwar auch in einem statischen Bereich eingesetzt werden, doch existieren für einen solchen Fall effizientere Lösungsverfahren.

Nachfolgend werden mehrere konkrete Anwendungen für das MoPiDiG Framework vorgestellt.

1.1.2.1 Public Transport on Demand

An hoch frequentierten Orten wie Bahnhöfen oder Marktplätzen ist es mit Hilfe von mobilen Endgeräten möglich, Personen anhand eines Profils zu gruppieren. Eine Gruppe zeichnet sich beispielsweise durch die gemeinsame Nutzung eines Transportmittels aus. Da sich die Gruppe spontan findet, kann dabei auf einen festen Zeitplan verzichtet werden. Nach Bildung der Gruppe wird hierfür ein entsprechendes Transportmittel geordert. Je nach Größe der Gruppe kann dies ein Taxi oder aber auch ein Bus sein. Weil sich in diesem Transportmittel nur Fahrgäste mit ähnlichen Zielen befinden, kann auf einen vordefinierten Streckenplan verzichtet und die Haltestellen gemäß den Wünschen der Passagiere dynamisch festgelegt werden.

Auf Grund der profilbasierten Routenbestimmung werden die Fahrgäste schneller an ihr gewünschtes Ziel transportiert. Durch das gezielte Finden von Gruppen für die Transportmittel wird außerdem die Wartezeit für die Fahrgäste auf dieses Transportmittel reduziert. Durch Verwendung der lokalen Kommunikationsschnittstelle entstehen dem Nutzer auch keine zusätzlichen Kosten. Seitens der Transportmittelbetreiber sind durch Umsetzung obigen Systems Einsparmaßnahmen zu verzeichnen. Zum einem können die Transportmittelbetreiber ihre Transportmittel entsprechend dimensionieren. Dies führt zu einer verbesserten Auslastung. Zum anderen reduzieren sich die Betriebskosten (z. B. Treibstoffkosten) der Transportmittel auf Grund der optimierten Route. Das Transportmittel ist außerdem schneller wieder für einen neuen Einsatz bereit [BS03].

In einem Beispiel - welches im Kapitel 5 im Detail erläutert wird - beschränken sich die Transportmittel auf Taxis. Die potenziellen Passagiere befinden sich an einem Flughafen oder Bahnhof. Während die Nutzer des Systems auf ihre Gepäckstücke warten oder sich zum Ausgang bewegen, erfolgt der Gruppierungsvorgang. Ergebnis ist eine Gruppe mit bis zu vier Personen, die zusammen ein Taxi benützen können und sich die Kosten hierfür untereinander aufteilen.

1.1.2.2 Profilbasierte Führungen

In Museen oder anderen kulturellen Einrichtungen werden Führungen angeboten, in denen ein Experte Details über die Exponate erklärt. In den seltensten Fällen ist solch eine Führung auf die Teilnehmer abgestimmt. Dies bedeutet, der Inhalt der Führung ist

unabhängig von der Teilnehmerzusammensetzung und den Teilnehmerinteressen. Doch gerade letztere divergieren häufig. Mit dem MoPiDiG Framework ist es möglich, Führungen nach den individuellen Wünschen der Teilnehmer abzuhalten oder potenzielle Teilnehmer in Gruppen zu formieren, die ähnliche Hauptinteressen verfolgen. Die Gruppierung kann bereits bei der Anfahrt (z. B. in öffentlichen Bussen oder an der Kasse) erfolgen, ist aber auch noch bis unmittelbar vor Führungsbeginn möglich. Auf diese Art kann die Qualität der Führungen verbessert werden [BS03].

1.1.2.3 Dynamisches Gruppieren im Fertigungsumfeld

In Produktionsanlagen oder Fertigungsstätten werden zum Zwecke der optimalen Zusammenarbeit oft dynamische Gruppen benötigt. Das Profil eines Arbeiters besteht aus seinen Fähigkeiten und Präferenzen. Auch Arbeitszeiten und Ruhepausen sind somit Teil des Profils. Soll ein bestimmtes Werkstück gefertigt werden, so wird mit diesen Profildaten die Gruppe bestimmt, die für den Auftrag am besten geeignet ist.

1.1.2.4 Weitere Anwendungen

Im Bereich des mobilen Entertainments sind noch weitere Anwendung des MoPiDiG Frameworks denkbar. So erfreuen sich Spiele auf mobilen Endgeräten zunehmender Beliebtheit. Mit dem MoPiDiG Framework könnten für Gruppenspiele die passenden Spieler innerhalb einer Menschenmenge gefunden werden. Die Gruppierung könnte beispielsweise hinsichtlich des Genres erfolgen.

1.2 Gliederung der Arbeit

In Kapitel 2 werden aktuelle Verfahren aus der Informationstechnologie vorgestellt. Die Begriffe Ubiquitous Computing, Pervasive Computing und Ambient Intelligence werden definiert und voneinander abgegrenzt. Ferner wird mit jedem Begriff eine Verknüpfung zum MoPiDiG Framework hergestellt.

Kapitel 3 beschäftigt sich mit der Einordnung der Arbeit in das wissenschaftliche Umfeld. Für die wichtigsten Teilgebiete der Arbeit wird beschrieben, welche Verfahren bereits existieren und wie diese für die Entwicklung des MoPiDiG Frameworks verwendet werden können.

Es werden mobile ad hoc Netzwerke vorgestellt und deren Architektur und Funktionsweise erläutert, sowie ein mathematisches Modell für ad hoc Netzwerke präsentiert. Anschließend werden existierende Arbeiten zu Benutzerprofilen präsentiert. Der Focus ist Literatur zur Darstellung von Profilen und Möglichkeiten der Generierung von Profilen. Danach werden Methoden der Gruppenbildung und Clustering aus verschiedenen Gebieten vorgestellt. Abschließend wird Literatur über Anwendungen dargeboten, die sich mit Gruppenbildung in ad hoc Netzwerken beschäftigen.

Kapitel 4 beschreibt den allgemeinen Aufbau des MoPiDiG Frameworks. Neben dem eigentlichen Gruppierungsvorgang sind noch weitere Mechanismen notwendig. Hierzu zählen die Segmentierung eines ad hoc Netzwerkes und die optionale Überlagerung eines Netzwerks mit einer Sekundärstruktur, welche beide aus Effizienzgründen durchgeführt werden.

Um den Zusammenhang der einzelnen Komponenten zu beschreiben, wird zuerst die Architektur des MoPiDiG Frameworks präsentiert und anschließend jede einzelne MoPiDiG Komponente im Detail erläutert. Abschließend wird noch der Verlauf einer Gruppierung beschrieben.

Ist das Ziel von Kapitel 4 eine domänenunabhängige Beschreibung von MoPiDiG, so steht in Kapitel 5 eine konkrete Anwendung im Vordergrund. Hier wird das Taxi-Sharing-Szenario näher untersucht. Hierbei wird die entwickelte Simulationsumgebung vorgestellt und es wird verdeutlicht, wie sämtliche in Kapitel 4 erwähnten Verfahren für das Szenario konfiguriert werden müssen. Als weiteres Beispiel wird MoPiDiG im Bereich der profilbasierten Führungen angewendet.

Kapitel 6 präsentiert und interpretiert Simulationsergebnisse des Taxi-Sharing-Szenarios. In diesem Kapitel wird u. a. ermittelt, bis zu welcher Teilnehmergeschwindigkeit das MoPiDiG Framework verwendet werden kann. Ferner wird ein Zusammenhang zwischen der Gruppengröße und der Teilnehmergeschwindigkeit ermittelt. Auch die Anzahl an benötigten Nachrichten wird näher betrachtet.

Mit Hilfe mathematischer Verfahren werden diese Simulationsergebnisse überprüft.

Kapitel 7 fasst sowohl den Aufbau des MoPiDiG Frameworks als auch die Ergebnisse der Arbeit zusammen und führt eine Bewertung durch. Abschließend erfolgt ein Ausblick, in dem Möglichkeiten der Erweiterung des MoPiDiG Frameworks angegeben werden.

Kapitel 2

Grundlagen

Seit Anfang der 90er Jahre wurde eine Vielzahl an Begriffen für neue Trends in der Informationstechnologie eingeführt. Einen Überblick über die Kreativität der Entwickler gibt Abbildung 2.1, wobei diese Ansammlung noch weit von der Vollständigkeit entfernt ist.

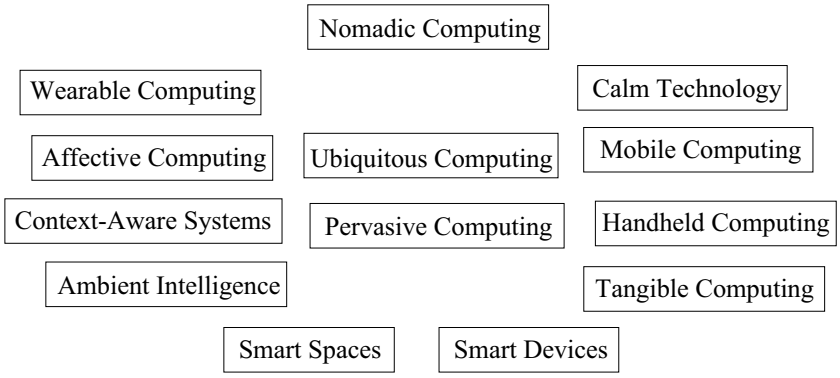


Abbildung 2.1: Begrifflichkeiten

Die Einführung dieser Begriffe führte teilweise zu Verwirrungen, da verschiedene Ausdrücke für ein und denselben Technologiebereich geschaffen wurden. Andere Bezeichnungen wiederum differieren nur marginal von bereits existierenden, so dass deren Einführung nur bedingt gerechtfertigt ist.

Aus der Fülle der Begriffe in Abbildung 2.1 wurden *Ubiquitous Computing*, *Pervasive Computing* und *Ambient Intelligence* ausgewählt, da sich diese in der Branche etabliert haben. Dieses Kapitel klärt diese Ausdrücke und knüpft eine Verbindung zwischen MoPiDiG und dem jeweiligen Technologiebereich.

2.1 Ubiquitous Computing

In diesem ersten Abschnitt wird die Vision des *„Ubiquitous computing“* beschrieben, so wie sie Mark Weiser - Wissenschaftler bei Xerox PARX - zu Beginn der ’90er Jahre des

letzten Jahrhunderts formulierte.

Ausgehend vom Versuch einer Begriffsklärung wird nachfolgend beschrieben, wie die Vision Realität werden kann. Hierzu wird die Evolution hin zu Ubiquitous Computing erläutert. Die notwendigen Voraussetzungen für die Entwicklung werden anschließend diskutiert.

2.1.1 Definition

Der Begriff *Ubiquitous Computing*[1] beschreibt die Allgegenwärtigkeit von kleinsten, miteinander drahtlos vernetzten Rechengeräten. Entscheidend ist, dass sie quasi unsichtbar in beliebige Alltagsgegenstände integriert sind oder an diese angeheftet werden können.

Der Begriff wurde von Mark Weiser in seinem wegweisenden Artikel „The Computer of the 21st century“ [Wei91] geprägt. Dieser beginnt mit den Worten:

> *„The most profound technologies are those that disappear. They wave themselves into the fabric of everyday life until they are indistinguishable from it.“*

Weiser deutet bereits mit diesem Satz an, welches Vorhaben er anstrebt. Ziel ist es, sämtliche Arten von Computertechnologie „verschwinden“ zu lassen, d. h. aus dem direkten Bewusstsein der Anwender zu bringen. Kleinste, miteinander kommunizierende Rechner wird es im Überfluss geben. Dennoch werden sie nicht mehr wahrgenommen, weil sie in Alltagsgegenstände eingebettet werden und mit unserer alltäglichen Umgebung verschmelzen. Er verdeutlicht dies mit folgender Aussage:

> *„Ubiquitous computing has as its goal the enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user.“* [Wei93b]

Eine andere Definition stammt von Gellersen [Gel99]. Er kennzeichnet Ubiquitous Computing durch zwei primäre Tatsachen. So lassen sich zum einem aufgrund der zunehmenden Miniaturisierung Geräte überall mithinnehmen und gewähren dadurch den uneingeschränkten Zugriff auf spezielle Dienste. Zum anderen verschmelzen beim Ubiquitous Computing Informationstechnologien mit der räumlichen Umgebung, sodass sie überall vorhanden sind.

2.1.2 Entwicklung

In einem späteren Artikel beschreiben Weiser und Brown [WB97], auf welche Weise sich die Entwicklung hin zu Ubiquitous Computing vollziehen kann.

In den frühen Tagen der rechnergestützten Informationsverarbeitung dominierte die Mainframe[2] Ära. Zu dieser Zeit mussten sich viele Benutzer solch einen Großrechner teilen, weil die Kosten hierfür enorm waren. Der Zugriff auf ein Mainframe erfolgte durch so genannte Terminals. Ein Terminal ist eine Kombination aus Bildschirm und Tastatur ohne eigenen Massenspeicher (z. B. Festplatte) und lediglich soviel Hauptspeicher, wie erforderlich ist, um den Bildschirminhalt darzustellen. Der Vorteil dieser Terminals liegt

[1] Der deutsche Ausdruck hierfür wäre in etwa allgegenwärtige (= ubiquitäre) Informationverarbeitung
[2] Mainframe (engl.) = Großrechner

darin, dass der Anwender an jedem Terminal arbeiten kann und immer seine Benutzerumgebung vorfindet. Da die Terminals über ein gewisses Areal verteilt sein können, kann in diesem Zusammenhang von einer ersten Art von *Mobile Computing* gesprochen werden. Ein Benutzer kann jedes Terminal verwenden und er findet stets seine gewohnte Arbeitsumgebung vor. In dieser ersten Art *Mobile Computing* ist somit nur der Benutzer mobil. Durch eine ausreichende Abdeckung mit Terminals ist jedoch ein Arbeiten an fast jedem Ort möglich.

Durch technologische Fortschritte im Hardware-Sektor (z. B. Miniaturisierung von Prozessoren) wurde die Entwicklung kleinerer Geräte realisierbar. Deshalb wurde es mit der Zeit möglich, dass die Anwender an eigenständigen Rechnern arbeiteten. Nach und nach konnten sich auch Privatanwender solch ein Gerät auf den Schreibtisch stellen. Diese Art Rechner wurde somit zum *„Personal-Computer*[3]“ (PC), weswegen diese Periode *PC-Ära* genannt wird. Doch warum der Terminus *Personal-Computer*? Der offensichtliche Punkt ist der, dass es ein zur persönlichen Verwendung geeignetes und auf eigene Bedürfnisse anpassbares Computersystem ist. Der Rechner wird personalisiert, d. h. er wird mit Applikationen, Hintergrundbildern, etc. so angepasst, dass er den Wünschen seines Besitzers gerecht wird. Diese Personalisierung kann sich durchaus zur Personifikation steigern, wenn z. B. eine Namensgebung erfolgt. Da der Preis anfänglich nicht unerheblich war, entwickelte der PC sich auch zum Statussymbol. Aus diesen Gründen teilt der Eigentümer „seinen“ PC auch nur ungern mit anderen Personen.

Technologisch waren die PCs ebenfalls eine Herausforderung: Im Gegensatz zum Großrechner muss ein PC mit Komponenten verschiedenster Hersteller verlässlich arbeiten, wobei die Komponenten typischerweise aus der billigen Serienproduktion stammen. Eine Unmenge an Software wurde und wird entwickelt. Wobei hierbei zu bedenken ist, dass auch der Software-Entwicklungsprozess die Herstellerunterschiede berücksichtigen muss, damit eine funktionsfähige Interoperabilität der Komponenten gewährleistet ist.

Bisher wurden Mainframes und PCs nur als einzeln vorhandene Rechner betrachtet, die mit keinem anderen Rechner verbunden waren. Doch auch dieser Schritt vollzog sich nach und nach. Zuerst wurden Großrechner der Universitäten miteinander verbunden und bildeten somit die erste Form des Internet. Das Internet als weltweites Rechnernetz bietet Dienste wie E-Mail, Diskussionsgruppen (Newsgroups), Chats, Dateiarchive etc. Seine große Bekanntheit verdankt das Internet jedoch dem World-Wide-Web. Um die steigende Anzahl an Rechnern zu verdeutlichen, zeigt die Tabelle 2.1 die Anzahl der Hosts im Internet seit 1979 [Int04]. Ab dem Jahr 2009 handelt es sich um zukünftige Daten, geschätzt auf Basis gleich bleibender Entwicklung.

Durch das Internet wurde es möglich, dass sich beliebige PCs miteinander verbinden, um so direkt miteinander zu interagieren.

Die Existenz von immer mehr vernetzten Computern führt nach Weiser unweigerlich zur *„Ubiquitous Computing Ära“*. Sie ist dadurch gekennzeichnet, dass es kleinste, miteinander kommunizierende Rechner im Überfluss geben wird. Dennoch werden sie nicht mehr wahrgenommen, weil sie in Alltagsgegenstände eingebettet werden und mit unserer alltäglichen Umgebung verschmelzen. Computer wird es in allen Größen geben - begonnen bei einem mikroskopisch kleinen Sensor bis zum noch immer existierenden Mainframe.

[3] Anm.: Bei dem Terminus *„Personal Computer“* handelt es sich um einen englischen Ausdruck, der mit *„persönlicher Computer“* übersetzt werden kann.

Jahr	Rechneranzahl	Verhältnis Rechner pro Person[4]
1979	188	
1989	130.000	
1999	56.218.000	100 Menschen / 1 Computer
2004	233.101.481	27 Menschen / 1 Computer
2009	15.180.000.000	1 Mensch / 2.25 Computer
2019	2.565.969.000.000	1 Mensch / 350 Computer

Tabelle 2.1: Anzahl an Rechnern im Internet

Der wesentlichste Schritt des *Ubiquitous Computing* - und bei einer solchen Menge an Rechengeräten auch ein unumgänglicher - ist das Verschwinden der Endgeräte aus dem Bewusstsein der Nutzer. Computer mutieren somit zum Gebrauchsgegenstand. Weiser akzentuiert dies durch

> *„A good tool is an invisible tool. By invisible, I mean that the tool does not intrude on your consciousness; you focus on the task, not on the tool.“* [Wei94]

Dies steht im strengen Gegensatz zum PC Zeitalter, als der PC im Zentrum des Interesses stand. Obwohl heutzutage neben den PCs auch bereits kleine Endgeräte wie z. B. PDAs[5] existieren, werden diese immer noch - unabhängig von ihrer Größe - wie PCs benutzt.

Um eine ubiquitäre Umgebung zu schaffen, schlägt Weiser als ersten Schritt eine Art „intelligente Büroumgebung“ vor. Als Geräte sollen so genannte Tabs (elektronische PostIts), Pads (elektronisches Papier) und Boards (interaktive Tafeln) eingeführt werden. In einem Büro sollten nach Weiser ([Wei93a]) für jede Person hunderte von Tabs, ein paar Dutzend Pads und ein oder zwei Boards existieren.

Solch eine Büroumgebung wurde bereits 1991 im Palo Alto Reseach Center der Firma Xerox kreiert, um das Arbeiten in einer solchen Umgebung zu erfahren, daraus zu lernen und weitere Schritte zu planen.

Seit den Anfängen von *Ubiquitous Computing* haben sich Teilmengen der Vision erfüllt. Die Entwicklung ständig kleiner werdender Endgeräte und permanenter Kommunikation ist bereits Realität. Da aber nicht nur technologische Mittel zu Weisers Vision notwendig sind, sonder auch ideologische, wird noch einige Zeit verstreichen, bis *Ubiquitous Computing* gemäß Weisers Vision vollständig verwirklicht ist. Anzeichen, dass die Vision Realität werden kann, sind in jedem Falle zur Genüge gegeben.

2.1.3 Voraussetzungen und Herausforderungen

In der Ubiquitous Computing Ära werden Computer so zahlreich sein, dass sie überall anzutreffen sind und gleichzeitig aus dem Focus der Aufmerksamkeit verschwinden. Um diese ehrgeizigen Ziele verwirklichen zu können, sind einige Voraussetzungen zu erfüllen.

[4] Die dritte Spalte stellt die Anzahl der Rechner im Internet der momentanen bzw. zukünftigen Weltbevölkerung gegenüber. Der eingetragene Wert ist somit ein Mittelwert, der regionale Unterschiede nicht berücksichtigt.

[5] PDA = Personal Digital Assistent. Hierbei handelt es sich um einen tragbarer Rechner in Notizbuchformat.

Die Myriaden an Rechnern müssen klein und unauffällig sein. Dies bedeutet, zukünftige Hardware muss sich weiter miniaturisieren.

Da ein wesentlicher Aspekt des Ubiquitous Computing die gleichzeitige Nutzung einer großen Anzahl an Rechnern sein wird, ist auch deren Vernetzung von extremer Bedeutung. Weiser [Wei91] schlägt drei Arten von notwendigen Verbindungen für den Datenaustausch vor: Für den Austausch großer Datenmengen wird eine verdrahtetes Hochgeschwindigkeitsnetz (z. B. Ethernet) benötigt. Ferner ist für die Mobilkommunikation ein drahtloses Netz über große Entfernung (z. B. GSM, UMTS) notwendig. Für die Kommunikation von Sensoren u. Ä. reicht ein drahtloses Netz über kurze Entfernung aus (z. B. Bluetooth, WiFi), da diese Geräte nur über geringe Sendeleistung verfügen.

Derzeit existieren zwar für jede Kategorie mehrere Technologien, eine allumfassende ist jedoch nicht in Sicht. Geräte müssen so beschaffen sein, dass eine Kommunikation in unterschiedlichen Netzen (z. B. sowohl GSM als auch WLAN) möglich ist. Ein Wechsel zum jeweils geeignetsten Netz hat automatisch erfolgen (vertikales Handover).

Eine garantierte Bandbreite zu gewährleisten - auch bei vielen Benutzern - ist vor allem bei mobilen Anwendungen schwierig. Wird mehr Bandbreite benötigt, so kann im verkabelten Fall die Anzahl der Glasfaserleitungen erhöht werden. Bei drahtlosen Netzen ist dies nicht möglich, da der Frequenzbereich nicht erweitert werden kann. Als einzige Alternative kann die Zellengröße reduziert werden. Dieser Vorgang erfordert jedoch erhebliche Investitionskosten und Umstrukturierungsmaßnahmen [Mil01].

2.1.4 MoPiDiG und Ubiquitous Computing

Nachdem die Ausführungen zu Ubiquitous Computing abgeschlossen sind, erfolgt abschließend noch eine Stellungnahme, ob und wie sich MoPiDiG in dieses Gebiet einordnen lässt.

Das MoPiDiG Framework stellt eine Alltagsanwendung dar, auf die nur schwer zu verzichten ist. Das stellt sicherlich eine nicht unwesentliche Gemeinsamkeit mit Anwendungen des Ubiquitous Computing dar. In der momentanen Entwicklung ist jedoch das mobile Endgerät noch zu präsent um der Vision gänzlich zu entsprechen, was sich jedoch in zukünftigen Versionen mit zunehmender Miniaturisierung ändern wird.

2.2 Pervasive Computing

In diesem Abschnitt wird der Begriff des *Pervasive Computing* näher betrachtet. War Ubiquitous Computing eine Vision und damit eher langfristig angelegt, wurde Mitte der neunziger Jahre primär von IBM der Ausdruck *Pervasive Computing* geprägt.

Pervasive Computing zielt mehr darauf ab, mit sofort einsetzbaren Technologien Lösungen zu erreichen, wobei Computertechnologie sukzessive ins Leben der Personen eindringt. Pervasive Computing kann aus diesem Grund mehr als Prozess aufgefasst werden und weniger als Vision.

2.2.1 Definition von Pervasive Computing

IBM definiert Pervasive Computing wie folgt:

> *„Pervasive computing is about making our lives simpler. Pervasive computing aims to enable people to accomplish an increasing number of personal and professional transactions using a new class of intelligent and portable devices. It gives people convenient access to relevant information stored on powerful networks, allowing them to easily take action anywhere, anytime. These new intelligent appliances or 'smart devices' are embedded with microprocessors that allow users to plug into intelligent networks and gain direct, simple, and secure access to both relevant information and services. These devices are as simple to use as calculators, telephones or kitchen toasters.“* [IBM02]

Während unter ubiquitären, überall verbreiteten Informationstechnologien i. A. Rechengeräte gemeint werden, deren Einsatz nicht an einen fixen Ort gebunden ist, meint der Begriff des Pervasive Computing genauer die Einbettung informationstechnologischer Anwendungen in Umgebungen und Abläufe mit dem Ziel, dass Informationstechnologie überall vorhanden ist.

Hansmann *et al.* [HMNS03] geben folgende Definition an:

> *„Convenient access, through a new class of appliances, to relevant information with the ability to easily take action on it when and where you need it“* .

War Weisers Vision aus Abschnitt 2.1.1 mehr benutzerfokussiert, indem er vorschlug, wie der Benutzer die Technik zu sehen hat, liegt der Schwerpunkt in Pervasive Computing mehr in der Entwicklung von *Appliances*. Dies sind Geräte, die für bestimmte Anwendungen entworfen werden müssen, um sich so besser in Abläufe einfügen zu lassen als Standardrechner.

Für Agoston *et al.* [AUN01] bedeutet Pervasive Computing schlicht:

$$\begin{matrix}\text{Anytime} \searrow \\ \text{Anywhere} \nearrow\end{matrix} \text{ Any Device} \rightarrow \text{Any Network} \rightarrow \text{Any Data}$$

Somit wird Pervasive Computing reduziert auf permanenten, ortsunabhängigen Zugriff auf irgendwelche Daten über ein beliebiges Netzwerk, der mit einem Gerät nach Wahl erfolgen kann. Diese Definition könnte auch für Ubiquitous Computing gelten. Dies lässt bereits erkennen, dass Autoren zwischen den beiden Begriffen nicht bzw. nur bedingt differenzieren. In letzter Zeit werden beide Ausdrücke auch mehr und mehr synonym verwendet.

2.2.2 Grundsätze des Pervasive Computing

Inhalt dieses Abschnitts sind die Prinzipien des Pervasive Computing. Dies sind Eigenschaften, die eine entsprechende Anwendung des Pervasive Computing erfüllen muss.

2.2.2.1 Dezentralisierung - Alle für einen

Der erste und wesentlichste Grundsatz des Pervasive Computing ist eine dezentrale Architektur. In einem solchen System soll keine zentrale Instanz vorhanden sein um Koordinationsaufgaben zu übernehmen. Pervasive Computing verteilt auf diese Weise die

Verantwortung auf viele kleine Geräte. Die Teilnehmer (Knoten) des verteilten Systems verfügen nicht a priori über globales Wissen. Nachrichtenaustausch erfolgt nur mit den direkten Nachbarn. Die Knoten haben auch keine Kenntnis über die Topologie des Netzwerkes oder die Anzahl der Knoten im verteilten System [MKDW03].

Der Verzicht auf eine Koordinationseinheit wirkt sich kontraproduktiv hinsichtlich der Steuerung des verteilten Systems aus. Um einen globalen Ablauf zu erreichen, sind Verfahren zur Entscheidungsfindung notwendig.

2.2.2.2 Diversifikation - Abwechslung ist alles

Der Standardrechner von heute gleicht einer Universalmaschine. Mit ihm ist es sowohl möglich einen Roboter in einer Produktionsanlage zu steuern, als auch einen Brief mit einem Textverarbeitungssystem zu verfassen oder zur Entspannung Spiele zu spielen.

Der Pervasive Computing Ansatz versucht eine Abkehr von dieser Art Universalmaschine. Stattdessen sollen viele unterschiedliche Geräte existieren, die jeweils hoch spezialisierte Aufgaben übernehmen.

Da diese Geräte über unterschiedliche Ressourcen und Fähigkeiten verfügen (z. B. unterschiedliche Bildschirmgröße, differierende Prozessorleistung), besteht die primäre Herausforderung darin, dass Anwendungen sich diesen unterschiedlichen Gegebenheiten automatisch anpassen.

2.2.2.3 Konnektivität - Die Verbindung zählt

In einer Pervasive Computing Umgebung sind alle Geräte miteinander verbunden. Ein PDA kann mit einem Mobiltelefon kommunizieren, welches wiederum Daten mit dem Internet austauscht. Hansmann *et al.* [HMNS03] drücken dieses Prinzip folgendermaßen aus:

> *„Everybody's software, running on everybody's hardware, over everybody's network."*

Eine Anwendung ist somit mit einem Höchstmaß an Heterogenität konfrontiert. Um diese globale Interoperabilität zu erreichen, ist die Etablierung von Standards unumgänglich.

2.2.2.4 Einfachheit - Weniger ist mehr

Der bereits erwähnte Universalrechner hat bezüglich seiner Benutzung Nachteile. Er ist zu komplex in seiner Zusammensetzung und kompliziert zu benutzen. Um ein neues Problem mit dieser Maschine zu lösen ist eine längere Einarbeitungsphase notwendig. Selbst wenn diese Phase bereits abgeschlossen ist, treten häufig noch Probleme auf.

Satyanarayanan [Sat04] begründet die Notwendigkeit zur Einfachheit durch Vergleich:

> *„The most successful technologies have low usage Complexity in spite of substantial internal Complexity."*

Durch die Ablösung der Universalmaschine durch viele spezialisierte Geräte soll die Bedienung und Verwendung erleichtert werden. Ähnlich wie zur Benutzung eines Bleistiftes

keine Bedienungsanleitung benötigt wird, soll es mit Geräten des Pervasive Computing sein.

Hierbei darf das Attribut „einfach“ nicht mit dem Begriff „primitiv“ gleichgesetzt werden. „Einfachheit“ bedeutet in diesem Zusammenhang, dass eine komplexe Technologie von einer leicht und intuitiv zu bedienenden Mensch-Maschine-Schnittstelle umgeben ist.

2.2.3 Entwicklungschritte zu Pervasive Computing

Wurden im letzten Abschnitt die grundsätzlichen Eigenschaften von Pervasive Computing beschrieben, wird nun eine Möglichkeit aufgezeigt, wie sich Pervasive Computing aus bestehenden Technologien entwickelt hat bzw. weiterentwickeln kann.

Pervasive Computing ist nach der Entwicklung von verteilten Systemen mit Beginn der 1970er und Systemen des Mobile Computing anfangs der 1990er der nächste Schritt in dieser Evolution.

Wie Abbildung 2.2 zeigt, sind *verteilte Systeme* als Ausgangspunkt der Entwicklung zu

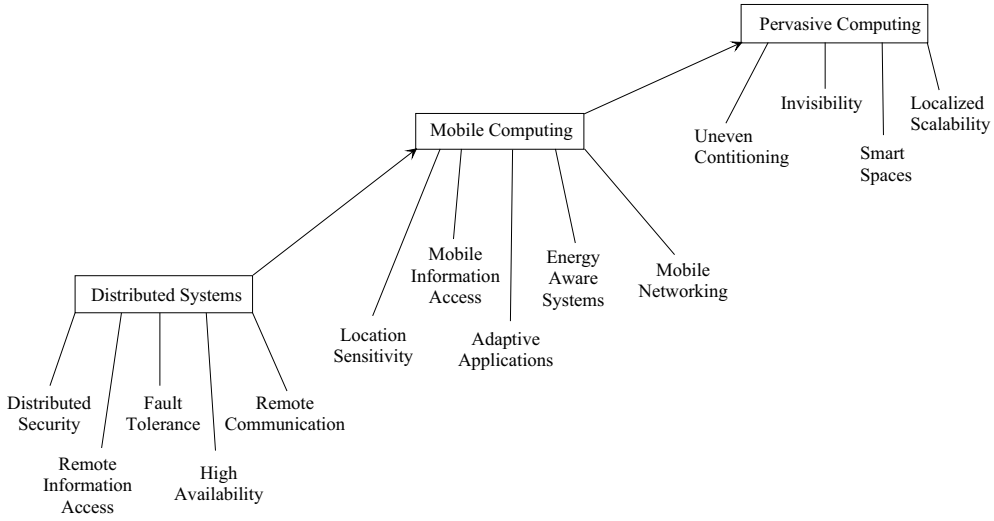


Abbildung 2.2: Die Entwicklung hin zu Pervasive Computing [Sat01]

Pervasive Computing zu sehen. Ein verteiltes System ist ein Zusammenschluss unabhängiger Computer, welches sich für den Benutzer als ein einzelnes System präsentiert. Um Rechner zusammenzuschließen müssen sie physikalisch miteinander verbunden werden, damit sie auf diese Weise miteinander kommunizieren können. Die Basis für eine Kommunikation war die Entwicklung von Kommunikationsprotokollen - das sind Regeln wie eine Kommunikation abläuft. Können Rechner Daten austauschen, so ist es auch möglich, dass Programme über mehrere Rechner verteilt ablaufen. Dabei ist ein wesentlicher Aspekt, dass dies für den Benutzer solcher Systeme transparent erfolgen muss.

Um mit verteilten Systemen verlässlich arbeiten zu können mussten auch Verfahren für den Fall entwickelt werden, wenn ein Element des verteilten Systems ausfällt. Als Beispiel seien nur Mechanismen wie Replikation oder Deadlock-Erkennung genannt.

Ein wesentlicher Punkt, den es zu erreichen gibt, ist die Synchronisation in einem verteilten System. Da es in einen solchen System viele Ressourcen gibt, auf die aber nur einer zugreifen darf (z. B. Drucker), sind solche Methoden nötig.

Waren verteilte Systeme an einen fixen Ort gebunden und über Kabel miteinander verbunden, wurde dieses Konzept im *Mobile Computing* um mobile Endgeräte mit drahtloser Verbindung erweitert. Dies brachte jedoch neue Probleme mit sich: weniger Bandbreite und eine langsamere Kommunikation. Durch die Mobilität wird eine höhere Fehlertoleranz nötig, da die Tendenz zu Verbindungsfehlern zunimmt.

Protokolle für verdrahtete Netzwerke müssen für den Einsatz in drahtlosen Netzwerken adaptiert werden. Ferner sind Mechanismen notwendig, die andere Geräte, die sich in Übertragungsreichweite befinden, entdecken und deren Dienste ermitteln.

Mobile Systeme sind durch einen restriktiven Umgang mit der Ressource Energie gekennzeichnet. Mobile Geräte verfügen über keine permanente stationäre Stromversorgung, sondern i. A. über eine wieder aufladbare Batterie. Deshalb passen sich beispielsweise Taktrate von Prozessor und Helligkeit des Displays den aktuellen Notwendigkeiten an.

Als eine „technology that disappears“ subsumiert *Pervasive Computing* sowohl das Konzept der verteilten Systeme als auch das des Mobile Computing, wobei es mit zusätzlichen Eigenschaften angereichert wird.

Im Pervasive Computing ist die Erzeugung und effiziente Nutzung von *Smart Spaces* [Sat04] ein wichtiger Bestandteil. Ein *Smart Space* ist ein abgegrenztes Areal, in welchem mit Hilfe zusätzlicher Infrastruktur Komponenten des Areals mit mobilen Geräten kommunizieren. Auf diese Weise wird es möglich, dass der Smart Space die Endgeräte manipuliert und die Endgeräte den Smart Space. Deswegen ist das Verhalten des Smart Space abhängig von der Anzahl und Art der in ihm befindlichen Personen.

Um ein Beispiel zu nennen, könnten in einem Smart Space die Temperatur und der Lichteinfall durch mobile Geräte gesteuert werden. Der Raum könnte Mobiltelefone dahingehend manipulieren, dass sich diese im Raum lautlos verhalten. Auch die Software eines sich im Smart Space befindlichen Geräts kann sich je nach Situation unterschiedlich verhalten.

Verschiedene Smart Spaces können sich hinsichtlich ihrer Infrastruktur unterscheiden, indem andere Sensoren und Geräte zur gegenseitigen Manipulation vorhanden sind. Auch können die Smart Spaces einen unterschiedlichen Grad an „smartness“ aufweisen. So wird ein hoch frequentierter Konferenzraum über mehr Infrastruktur verfügen als ein gewöhnliches Klassenzimmer.

Ein weiterer Punkt, den eine Pervasive Computing Umgebung erfüllen muss, ist das Verschwinden der Technologie aus dem Bewusstsein der Nutzer, wie es auch schon beim Ubiquitous Computing der Fall war. Eine nahtlose Integration der Soft- und Hardware in die Umgebung führt zu einer minimalen Benutzerablenkung, wodurch eine höhere Produktivität erreicht wird.

Ein pervasives System verhält sich so, wie es der Nutzer erwartet. Nur in den nötigsten Fällen wird ein Eingriff des Benutzers erforderlich, der aber in einer Art unterbewusster Handlung abgearbeitet werden kann.

Im Pervasive Computing bekommt der Term „Skalierbarkeit“ eine vielschichtigere Bedeutung. In bisherigen Client/Server Systemen ist eine Obergrenze an Anfragen bekannt,

die rechnerisch bewältigt werden kann. Da in einem Pervasive Computing System quasi alles miteinander vernetzt ist, kann keine Obergrenze an Geräten angegeben werden. Um dennoch mit einem mobilen Gerät arbeiten zu können, müssen die mit ihm verbundenen Geräte, die sich in ummittelbarer Nähe befinden, bevorzugt bedient werden, d. h. es werden primär nur Rechner berücksichtigt, deren Entfernung eine obere Grenze nicht überschreiten. Hiermit wird die Skalierbarkeit eine Funktion der Distanz. Deswegen spricht Satyanarayanan [Sat01] von „Localized Scalability[6]“ .

Durch verschiedene Interaktionsmöglichkeiten in unterschiedlich „smarten“ Umgebungen wird dem Benutzer das Vorhandensein und Nicht-Vorhandensein von Technologie bewusst. Deswegen wird in einem Pervasive Computer System ein Mechanismus benötigt, der im Stande ist, unterschiedliche Gegebenheiten auszublenden (masking uneven conditions). Eine Möglichkeit dies zu erreichen ist, dass Funktionalität, die der Raum nicht zur Verfügung stellt, vom Gerät kompensiert wird - was nicht immer durchführbar sein wird. Trifft ein mobiles Gerät andererseits auf Funktionen, die dem Benutzer unbekannt sind oder ihn überfordern würden, können diese ignoriert werden.

2.2.4 Gefahren und Probleme mit Pervasive Computing

Pervasive Computing soll den Menschen eine Erleichterung im täglichen Leben bringen [HMNS03]. Doch sind mit einem Fortschreiten der Technologie auch erhebliche Gefahren und Probleme verbunden. Mögliche gesundheitliche Einwirkungen werden schon länger in den Medien kontrovers diskutiert, wobei ein endgültiges Ergebnis noch nicht vorliegt.

Eine andere Gefahr entsteht durch die Tatsache, dass an allen Orten Informationstechnologie vorhanden und eine permanente Kontrolle von Personen möglich ist. So ist bei Cunningham & Cunningham [Cun04] zu lesen:

> *„Pervasive Computing means pervasive surveillance.“*

So kann der Aufenthaltsort (z. B. mittels Cell-Id der Mobilfunknetze, Zugangskontrolle mit Hilfe von Smart Labels) von Personen festgestellt werden. Erfolgt dies permanent, entsteht ein Bewegungsprofil. Ergebnis ist der Verlust der Privatsphäre.

Doch nicht nur technische Gefahren können auftreten, sondern auch ethische Probleme dürfen nicht außer Acht gelassen werden. Stone [Sto03] weist auch auf die Gefahr der Abhängigkeit des Menschen von der Technik hin. Setzen sämtliche alltägliche Vorgänge ein Funktionieren der Technik voraus, ist bei deren Ausfall ein Chaos zu erwarten. Werden ferner mehr und mehr Entscheidungen von technischen Geräten getroffen, ist bei einer falschen Entscheidung nur schwer ein Schuldiger auszumachen.

Zuletzt sollen noch kurz umweltpolitische Gesichtspunkte betrachtet werden. Durch Myriaden an Geräten wird der Stromverbrauch steigen, wovon die kleinen mobilen Endgeräte nur einen marginalen Teil verbrauchen werden und der Großteil für die sich im Hintergrund befindliche Infrastruktur benötigt wird. Laut Hilty *et al.* [HKS+03] kann durch die Vielzahl an Geräten auch ein Entsorgungsproblem entstehen, wenn die Halbwertszeit technischer Produkte auf Grund stetiger Neuentwicklungen weiter sinkt (es sei nur als Beispiel das Mobiltelefon genannt).

[6] ortsgebundene Skalierbarkeit

2.2.5 MoPiDiG und Pervasive Computing

Auch am Ende dieses Abschnitts erfolgt eine Einordnung von MoPiDiG in die entsprechende Technologie.

Da Pervasive Computing weniger eine Vision darstellt und mehr auf eine konkrete Realisierung abzielt, lässt sich MoPiDiG fast uneingeschränkt mit dem weiten Gebiet des Pervasive Computing verbinden. Mit dem MoPiDiG Framework ist es möglich, verteilte, mobile Anwendungen auf kleinen Endgeräten zu entwickeln. Hiermit sind bereits drei Anforderungen für eine Pervasive Computing Anwendung erfüllt. Das Einfachheitskriterium ist jedoch zum momentanen Zeitpunkt nur bedingt erfüllt, da Konfigurationsarbeit geleistet werden muss. Dies bezieht sich sowohl auf die Entwickler- als auch auf die Benutzersicht. Beide Gruppen müssen das Framework bzw. die dazugehörige Anwendung hinsichtlich ihrer Verwendung anpassen.

2.3 Ambient Intelligence

In diesem Abschnitt soll der Begriff *Ambient Intelligence* näher beleuchtet werden. Bevor eine Begriffsklärung erfolgen kann, wird der Begriff des Kontexts eingeführt, der bei Ambient Intelligence von zentraler Bedeutung ist. Eine stufenweise Entwicklung hin zu Ambient Intelligence bildet den Abschluss dieses Abschnittes.

2.3.1 Context Aware Computing

Durch die Entwicklung tragbarer Computersysteme und die damit verbundene Mobilität der Benutzer wird ein neues Anwendungsspektrum möglich. Mobile Endgeräte können in verschiedenen Umgebungen (z. B. Büro, Fußgängerzone) verwendet werden. Die Information, die diese Umgebungen beschreibt, wird als Kontext bezeichnet. Dey [Dey01] definiert den Kontextbegriff so:

> *„Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves“.*

Nach obiger Definition ist jede zum Zeitpunkt einer Interaktion zur Verfügung stehende Information eine Kontext-Information.

Die Fülle der möglichen Kontextinformationen wird in vier Klassen eingeteilt [LS00]. *Aktionsbasierter Kontext* ist dadurch charakterisiert, dass der Computer beim Ablauf einer Applikation berücksichtigt, was der Benutzer sagt, worauf er schaut, womit er arbeitet etc. Beim *emotionellen Kontext* fließt in die Programmdurchführung ein, wie der Benutzer sich fühlt. Soll der bereits gespeicherte Kontext bei der Programmverarbeitung berücksichtigt werden, spricht man von *historischem Kontext*. Schließlich gibt es noch den physikalischen Kontext. Hierbei werden Daten über Ort, Zeit, aktuelles Datum und Identität des Benutzers bei der Applikationsdurchführung mit verarbeitet. Bei ortsabhängigen Anwendungen spielen insbesondere der aktionsbasierte und der *physikalische Kontext* eine Rolle [SAW94].

Chen und Kotz [CK00] unterscheiden zusätzlich noch *aktiven* und *passiven Kontext*. Unter *aktivem Kontext* wird die Teilmenge des Kontexts verstanden, die für die momentanen Applikationen relevant ist. Die aktuell nicht benötigten Kontextinformationen werden demgemäß als *passiver Kontext* bezeichnet.

Nach obiger Definition kann „Context Awareness“ als die Nutzung der Kontextinformationen durch ein System verstanden werden. „Context Aware Computing“ stellt ein Computersystem dar, das diese Daten aus seiner Umgebung ermittelt, interpretiert und benutzt, und dabei ggf. seine Funktionalität an die ermittelte Situation anpasst. Neben dem expliziten Input des Anwenders können somit auch noch verschiedenste Kontextelemente als zusätzliche Eingabedaten dienen. Abbildung 2.3 zeigt dies und beschreibt die

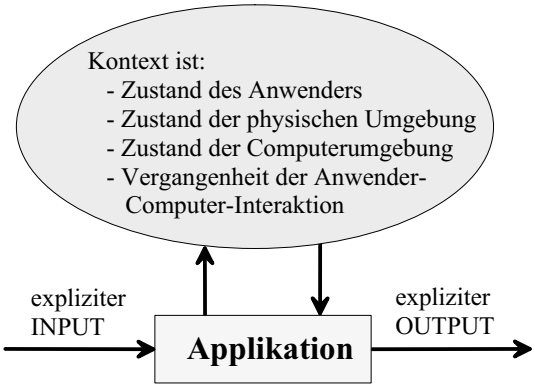


Abbildung 2.3: Context-Aware Computing nach der Definition von Lieberman und Selker [LS00]

möglichen Kontextelemente näher. Als Kontext wird hierbei alles außer dem expliziten Input und Output des Anwenders bezeichnet.

2.3.2 Definition von Ambient Intelligence

Nachdem das Context Aware Computing erläutert ist, kann näher auf den Begriff des *Ambient Intelligence* eingegangen werden.

Der Begriff *Ambient Intelligence* wurde 2001 von der Information Society Technologies Advisory Group (ISTAG) [DBS+01] ins Leben gerufen. Ziel war es Szenarios festzulegen, wie Menschen im Jahre 2010 ihr tägliches Leben verbringen.

Lindwer *et al.* [LMB+03] legen folgende Merkmale für *Ambient Intelligence* fest:

- unauffällige Hardware,
- Kommunikationsinfrastruktur für mobile und stationäre Geräte,
- dynamische und massive verteilte Netzwerke,
- natürliche Benutzerschnittstellen und
- Systemstabilität und Sicherheit.

Beruht eine Definition nur auf obigen Punkt, ist eine Abgrenzung zu Ubiquitous und Pervasive Computing nicht möglich.

Friedewald und Costa [FdC03] sprechen von einer Vision des *Ambient Intelligence*:

> *„The vision of Ambient Intelligence assumes a shift in computing from desktop computers to a multiplicity of computing devices in our everyday lives whereby computing moves to the background and intelligent, ambient interfaces to the foreground".*

Andere Definitionen benötigen den im letzten Abschnitt definierten Begriff des Kontexts. So ist nach Shadbolt [Sha03] *Ambient Intelligence* das Ergebnis einer Konvergenz aus folgenden Bereichen:

- Context Awareness,
- Ubiquitous oder Pervasive computing und
- Intelligent systems research.

Versteht Shadbolt [Sha03] unter Context Awareness den in Abschnitt 2.3.1 definierten Kontextbegriff, braucht es zu den zwei anderen Bereichen noch nähere Erläuterungen. Mit dem zweiten Punkt „Ubiquitous oder Pervasive computing" geht der Autor nicht davon aus, dass die zwei Konzepte bereits völlig umgesetzt sind. Es ist vielmehr nur ein erster Ansatz nötig. So sollen beispielsweise Möglichkeiten der spontanen drahtlosen Kommunikation vorhanden sein und eine Vielzahl an mobilen Endgeräten existieren. Unter *Intelligent systems research* nennt Shadbolt Begriffe wie die Entwicklung von Lernalgorithmen, Planung, Matchmaking, Spracherkennung, Sprachübersetzer, Gestikerkennung und -klassifikation. Viele dieser Konzepte stammen aus dem Gebiet der traditionellen Künstlichen Intelligenz [RN03], doch gehen manche darüber hinaus.

2.3.3 Entwicklungsschritte

Nach Berger *et al.* [Ber04; BMS05] kann Ambient Intelligence evolutionär über die Schritte Ubiquitous Communication und Context Awareness erreicht werden. Dies wird auch in Abbildung 2.4 verdeutlicht. Für den Beginn der Entwicklung ist eine Möglichkeit zur ubiquitären Kommunikation notwendig. Hiermit setzen Berger *et al.* voraus, dass Kommunikation mit mobilen Endgeräten bereits weit verbreitet ist und sich in der Gesellschaft etabliert hat. Durch zusätzliche Umgebungsinformationen wird der nächste Schritt „Context Awareness" erreicht. Durch intelligentes Verhalten (z. B. Adaptivität durch Lernalgorithmen) wird schließlich „Ambient Intelligence" ermöglicht. Gleichzeitig schlägt Berger noch ein Schichtenmodell zur Anwendungsentwicklung vor.

Die fünf Schichten - von der Geräte- bis zur Präsentationsschicht - sind in allen Entwicklungsschritten vorhanden und werden jeweils sukzessive erweitert. Dieses Schichtenmodell ermöglicht ein sehr evolutionäres Vorgehen, da Schicht für Schicht nacheinander - voneinander unabhängig - ausgebaut werden kann. Auch müssen nicht sämtliche Schichten eines Schrittes komplett implementiert sein, um bereits Anwendungen nutzen zu können. Die einzelnen Schichten werden nachfolgend für jeden Entwicklungsschritt gemäß Abbildung 2.4 beschrieben.

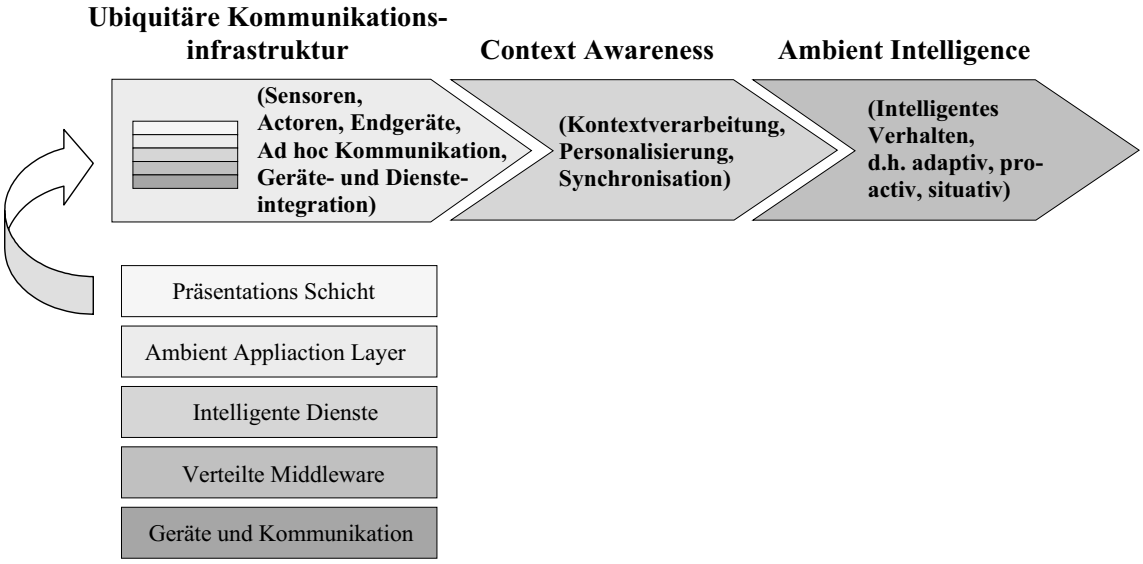


Abbildung 2.4: Ambient Intelligence nach Berger [Ber04]

2.3.3.1 Geräte und Kommunikation

Zu Beginn wird vorausgesetzt, dass mobile Kommunikation bereits vorhanden ist. Diese kann räumlich begrenzt (z. B. WLAN, Bluetooth) oder bereits ubiquitär sein (z. B. GSM, UMTS).

Im zweiten Schritt werden der Umgebung gemäß Abbildung 2.4 Sensoren und Aktoren zugefügt. Diese verfügen ebenfalls über eine Kommunikationsschnittstelle und sind miteinander vernetzt. Mobile Endgeräte können sich mit ihnen verbinden und Messwerte übermittelt bekommen oder Aktionen auslösen.

Die mobilen Geräte sind konfigurierbar hinsichtlich ihrer Kommunikationsmöglichkeiten (z. B. software defined radio).

2.3.3.2 Verteilte Middleware

Middleware ist eine Menge von allgemeinen Diensten, die zwischen der Systemplattform und den Anwendungen angesiedelt ist und deren Verteilung unterstützen.

Grundlage ist eine Dienste-orientierte Infrastruktur. Beispiel hier sind Web-Services. Benötigt wird außerdem noch eine Möglichkeit, Dienste anhand von benutzergerechten Beschreibungen zu finden und auf diese zuzugreifen.

Mit Peer-To-Peer Mechanismen können effiziente Zugriffsmethoden auf Dienste entwickelt werden. Mehrere Dienste können zu einem virtuellen komplexen Dienst vereint werden (Dienste-Komposition und Dienste-Orchestration). Geschieht diese Komposition anfänglich noch manuell, so kann sie nach und nach durch das Hinzufügen von Semantik in Form von Ontologien automatisch erfolgen. Eine Ontologie ist ein formal definiertes System von Dingen und Relationen zwischen diesen Dingen. Hierdurch ist es möglich, Dienste so zu beschreiben, dass sie automatisiert verwendet werden können.

2.3.3.3 Intelligente Dienste

Der zentrale Punkt sind die Intelligenten Dienste. Zu Beginn sind diese noch nicht vorhanden. Die komplette Nutzung wird vom Anwender gesteuert. Der Benutzer ist verantwortlich für die Suche und die richtige Bedienung des Dienstes.

Im nächsten Schritt wird gemäß Abbildung 2.4 dieser Vorgang automatisiert. Der Kontext gibt vor, welche Dienste benötigt werden, diese werden gesucht und genutzt. Dienste nutzen die verschiedenen Arten von Kontextinformationen, um hieraus weitere Informationen zu generieren (Reasoning). Dadurch, dass mehrere Personen gleichzeitig die gleichen Dienste nutzen, können Koalitionen entstehen. Im Gegensatz dazu kann aber um den simultanen Zugriff auf Dienste auch verhandelt werden.

Durch Verfahren wie Lernen und Planen wird schließlich adaptives und zielorientiertes Handeln ermöglicht. Die Anwendung lernt sich dem Benutzer anzupassen und handelt in seinem Interesse.

2.3.3.4 Ambient Application Layer

Diese Ebene stellt eine Art Assistenten-Rolle für den Benutzer dar. Der Assistent steuert die Anwendungen und die hierzu benötigte Dienste. Wird der Benutzer anfänglich noch des Öfteren für Aktionen benötigt, übernimmt der Assistent immer mehr Aufgaben.

Hierfür kennt der Assistent die Präferenzen „seines“ Benutzers. So kann der Assistent beispielsweise Termine für Meetings in seinem Auftrag vereinbaren. Der Assistent bemerkt Änderungen an den Präferenzen und handelt zukünftig dementsprechend. Auch ist es möglich den Kontext zu erweitern. Dies beschränkt sich nicht nur auf eine Aktualisierung der Kontextinformation, sondern auch auf die Einbindung neuer Kontextdaten.

2.3.3.5 Präsentations-Schicht

Diese Schicht beschreibt die Entwicklung der Benutzerschnittstellen. Können anfänglich Anwendungen nur mit gewöhnlichen Eingabemethoden (z. B. Tastatur) bedient werden, nehmen die Interaktionsmethoden laufend zu. Multimodale Eingabemöglichkeiten sind Kennzeichen des zweiten Schrittes. So kann ein Benutzer durch Sprache seine Anwendungen steuern. Doch auch die momentane Verfassung (emotionaler Kontext) und sein Standort gehen als implizite Parameter in die Anwendungen mit ein. Avatare[7] helfen dem Benutzer bei der Bedienung der Programme.

Ziel ist schließlich, dass nicht nur die Geräteumgebung, sondern auch die natürliche Umgebung (z. B. Wände, Tische) zur Präsentation und Benutzerinteraktion herangezogen wird.

2.3.4 MoPiDiG und Ambient Intelligence

Nach diesen Ausführungen wird noch betrachtet, wie MoPiDiG und das Technologiefeld Ambient Intelligence harmonieren.

Bereits an der Definition von Ambient Intelligence von Shadbolt lässt sich erkennen, dass MoPiDiG sämtliche dort erwähnten Kriterien erfüllt. In Ambient Intelligence nimmt der Kontextbegriff einen breiten Raum ein, was auch für MoPiDiG zutrifft, sofern der

[7] Ein Avatar ist eine Art künstliche Bildschirmgestalt

Transfer von Profil und Kontext vollzogen wird. Die geforderte allgegenwärtige Kommunikation wird durch ad hoc Netzwerke erfüllt. Wird das Gruppieren als intelligente Anwendung angesehen - was es auch ohne Einschränkung ist - sind schließlich alle Anforderungen erfüllt.

2.4 Zusammenfassung

Grundlage dieses einführenden Kapitels ist die Klärung aktueller Begrifflichkeiten sowie deren Vergleich.

Ubiquitous Computing ist eine Vision von Mark Weiser und beschreibt eine Welt mit Myriaden Rechengeräten, wobei diese Geräte durch ihre geringe Größe und intelligente Benutzungsmethoden aus dem Bewusstsein der Anwender verschwinden.

IBM führte Mitte der 90er Jahre den Begriff *Pervasive Computing* ein. Dieser Ausdruck will sich konkret von der Vision distanzieren und sich auf Anwendungen konzentrieren, die mit bereits existierenden Technologien umsetzbar sind.

Im Vordergrund von *Ambient Intelligence* steht die Intelligenz, die sich in den von dem Anwender verwendeten Zugangsgeräten, in einem Netzwerk, in den zugegriffenen Informationen oder der Umgebung manifestieren kann. Eine klare Abgrenzung zu den beiden anderen Begriffen ist nur bedingt möglich [Eik04].

Das MoPiDiG Framework kann primär mit jedem dieser Technologiebereiche in Verbindung gebracht werden. So lässt sich MoPiDiG ohne Zweifel als Anwendung der *Ambient Intelligence* betrachten. Auch die Anforderungen des *Pervasive Computing* werden erfüllt. Die Ansprüche von Weisers Vision an sind bei einer MoPiDiG Anwendung teilweise sichtbar.

Als Leslie Lamport - ein Pionier der Entwicklung verteilter Systeme - nach seiner Meinung gefragt wurde, was er über die Entwicklung neuer Begrifflichkeiten denke, antwortete dieser:

> *„These are the same old distributed systems, except with significantly different parameters. [..] I haven't found any good new theory coming from these new technologies; maybe someone else will.“* [Mil02]

Kapitel 3

Einordnung in die Literatur

Dieses Kapitel beschreibt bereits existierende Ansätze, die zur Lösung des MoPiDiG Frameworks verwendet werden können. Um eine MoPiDiG Anwendung zu entwickeln ist die Lösung dreier Teilproblematiken nötig, die bereits in Abschnitt 1.1 beschrieben werden.

Da die Gruppenbildung in einem mobilen Umfeld stattfindet, wird die Kommunikation unter den Teilnehmern sporadisch unterbrochen oder bricht eventuell sogar gänzlich ab. Deshalb sind Verfahren notwendig, die eine robuste Kommunikation gewährleisten. Wird zusätzlich die Anzahl der Kommunikationspartner auf entsprechende Größe reduziert, sinkt die Wahrscheinlichkeit eines Ausfalls und die Zeit, in der eine Gruppe gebildet werden kann, steigt an.

Eine weitere Herausforderung in einer MoPiDiG Anwendung ist das Finden von ähnlichen Daten und deren Zusammenschluss zu Gruppen. Hier werden Methoden aus dem Data Mining vorgestellt, deren Aufgabe es ist, aus einer Fülle von Datensätzen „ähnliche“ zu selektieren.

Ferner muss für ein MoPiDiG Szenario ein Profil für den Anwender existieren. Für dieses Profil muss eine interne Repräsentation definiert werden. In diesem Zusammenhang werden auch Mechanismen zur Profilerzeugung und Sicherheitsaspekte kurz betrachtet.

Bevor jedoch nachfolgend jede Teilproblematik näher beleuchtet und bereits existierende Literaturansätze präsentiert werden, werden die grundlegenden Konzepte von mobilen ad hoc Netzwerken erklärt, da sie die Basis eines MoPiDiG Szenarios bilden.

3.1 Mobile ad hoc Netzwerke

Tragbare Geräte wie PDAs, Mobilfunkgeräte oder Notebooks bieten dem Benutzer neue Möglichkeiten der Mobilität und eröffnen die Möglichkeit der drahtlosen Datenkommunikation. Eine spezielle Art von drahtlosen Netzen sind mobile ad hoc Netzwerke[1] die wenig bis keine Infrastruktur benötigen und bei denen sich die Netzwerktopologie spontan durch die Bewegung der Teilnehmer ändern kann.

[1] ad hoc ([lat. <zu diesem>], bildungssprachlich: zu diesem Zweck, eigens dafür; aus dem Augenblick heraus [Bro96]

3.1.1 Klassifikation von ad hoc Netzwerken

Mobile ad hoc Netzwerke (MANET) können anhand verschiedener Parameter, z. B. Größe des Netzes, Geschwindigkeit der Teilnehmer oder die verwendete Funktechnologie klassifiziert werden.

Im Folgenden werden zwei Arten von ad hoc Netzwerken auf Grund der verwendeten Infrastruktur klassifiziert. Prinzipiell können zwei Arten von ad hoc Netzwerken unterschieden werden.

3.1.1.1 Mobile ad hoc Netzwerke mit Infrastruktur

Die erste Möglichkeit ist ein infrastrukturbehaftetes ad hoc Netzwerk. Ein solches ist in Abbildung 3.1 dargestellt. Die Kommunikation zwischen den mobilen Endgeräten erfolgt

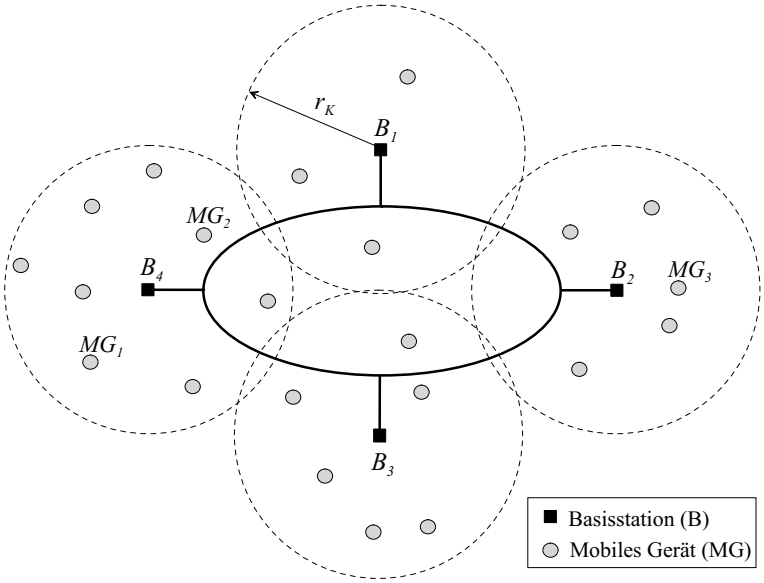


Abbildung 3.1: Ad hoc Netzwerk mit Basisstationen

über Basisstationen, die untereinander verbunden sind. Die Verbindung der Basisstationen kann sowohl drahtlos als auch verkabelt sein. Mit der Basisstation können Geräte kommunizieren, die sich innerhalb eines gewissen Radius r_k befinden. Die Kommunikation zwischen zwei Geräten erfolgt über die Basisstation und somit indirekt.

Will beispielsweise das Gerät MG_1 mit MG_2 kommunizieren, sendet MG_1 die Nachricht zuerst zur Basisstation B_4. Dieser ist bekannt, dass sich MG_2 in ihrem Zuständigkeitsbereich befindet und sendet somit die Nachricht weiter zu MG_2. Bei einer Kommunikation zwischen MG_1 und MG_3 muss B_4 die Nachricht an die anderen Basisstationen weiterleiten. Erhält B_2 diese Nachricht, stellt sie die Nachricht MG_3 zu. Die Übertragungsrate ist abhängig von der Anzahl der Benutzer, die Teil des Netzwerks sind, und von der Entfernung zur Basisstation.

Da die Benutzer der Geräte mobil sein können und somit vom Kommunikationsbereich einer Basisstation in den einer anderen gelangen können, muss die Verbindung von einer Basisstation auf eine andere übergeben werden. Dieser Vorgang wird als *Handover* bezeichnet.

Auf diese Weise können eine große Anzahl mobiler Systeme mit Hilfe einer relativ kleinen Anzahl statischer Basisstationen miteinander drahtlos kommunizieren.

3.1.1.2 Mobile ad hoc Netzwerke ohne Infrastruktur

Der Verzicht auf jedwede Infrastruktur bildete die Alternative zum vorhergehenden Abschnitt. Solch ein Netzwerk ist in Abbildung 3.2 dargestellt. Ein Gerät kann primär mit

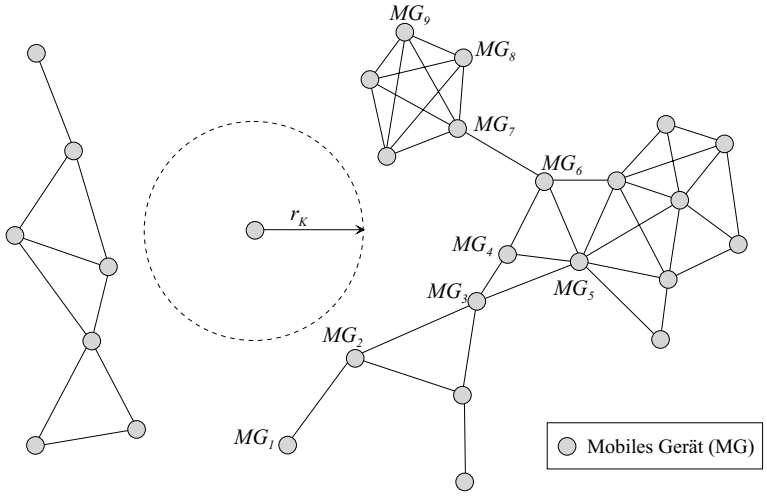


Abbildung 3.2: Ein ad hoc Netzwerk ohne zusätzliche Infrastruktur

all seinen direkten Nachbarn kommunizieren. Direkte Nachbarn sind hierbei jene Geräte, die sich innerhalb des Kommunikationsradius r_K befinden. Dadurch dass jedes Gerät aber noch als Router agiert und Pakete weiterleiten kann, können prinzipiell alle Geräte erreicht werden, zu denen ein Pfad existiert. So können in Abbildung 3.2 beispielsweise die Geräte MG_1 und MG_9 miteinander kommunizieren.

Zu bedenken ist allerdings, dass mit einer zunehmenden Anzahl an Geräten, die als Übermittler fungieren, die Stabilität der Kommunikation rapide abnimmt. Je mehr Geräte in diesen Vorgang involviert sind, desto wahrscheinlicher ist es auch, dass eines dieser Geräte während der Kommunikation ausfällt oder Verbindungen durch die Mobilität unterbrochen werden. Fällt beispielsweise das Gerät MG_4 aus, kann die Routingfunktion noch von MG_5 ausgeglichen werden. Bei einem Ausfall von MG_6 ist jedoch eine Kommunikation mit MG_9 nicht mehr möglich. Die Anzahl an Übermittlern wird häufig mit *Hop-Count* bezeichnet. Jedes Gerät reduziert den Hop-Count um den Wert eins. Ist dieser gleich Null, wird die Nachricht nicht mehr weitergeleitet.

Mobile ad hoc Netzwerke ohne Infrastruktur erlauben einen schnellen, kostengünstigen und unkomplizierten Netzwerkaufbau, wobei die maximale Teilnehmerzahl an Geräten

beschränkt ist und auch die Entfernung zwischen den einzelnen Computern nicht allzu groß sein darf - 30m ist etwa das Maximum bei einer Datenrate von 11Mb/s, was eigene durchgeführte Tests ergaben. Der Vorteil des ad hoc Modus ist, dass er sich jederzeit und überall einrichten lässt.

3.1.2 Kommunikationsprobleme in mobilen ad hoc Netzwerken

Durch die Mobilität der Geräte innerhalb eines ad hoc Netzwerkes treten Probleme auf, wobei sich die Tatsache, dass sich die Anwender bewegen, noch nicht negativ auswirken muss, sofern die Geschwindigkeitsvektoren gleich sind. Eine vorhandene Relativgeschwindigkeit wirkt sich jedoch nach gewisser Zeit auf die Stabilität der Kommunikationsverbindung aus.

Durch die Mobilität der Benutzer werden bestehende Kommunikationsverbindungen mit hoher Wahrscheinlichkeit nach einiger Zeit unterbrochen. Dies geschieht, indem sich Kommunikationspartner voneinander oder einer Basisstation entfernen. Dieser Vorgang kann permanent oder aber auch nur temporär sein. So kann ein entferntes Endgerät nach kurzer Zeit wieder mit der Basisstation oder dem Kommunikationspartner verbunden sein.

Ein mobiles Gerät kann auf Grund eines Defektes oder Strommangel ausfallen. Dieser Ausfall ist i. A. permanent, d. h. das Gerät wird in absehbarer Zeit nicht wieder als Teilnehmer des ad hoc Netzwerkes fungieren. Dieser Vorgang ist nicht vorhersehbar. Im Gegensatz dazu kann beim Ausschalten durch den Benutzer eine entsprechende Meldung an das ad hoc Netz erfolgen, wodurch die betroffenen Kommunikationspartner informiert werden können.

3.1.3 Routing

In Abschnitt 3.1.1.2 wurden infrastrukturlose ad hoc Netzwerke präsentiert. A priori ist eine Kommunikation in ad hoc Netzwerken nur möglich, wenn sich beide Kommunikationspartner innerhalb ihrer Kommunikationsradien befinden. Soll jedoch noch mit weiteren Teilnehmern kommuniziert werden (z. B. MG_1 mit MG_9 in Abbildung 3.2) ist ein Routingverfahren notwendig.

Die Aufgabe des Routings ist es, einen Pfad - wenn möglich den optimalsten - vom Sender zum Empfänger zu finden. Was für ein Pfad der beste ist, variiert von Fall zu Fall. Es kann beispielsweise der kürzeste Pfad sein, was durch Zählen der Übermittlungsknoten festgelegt wird. Der kürzeste Pfad muss allerdings nicht der schnellste sein, so dass auch die Geschwindigkeit ein Maß für die Güte sein kann. Solche Fragestellungen hinsichtlich des *Quality of Service* (Dienstgüte) und Routing werden von Chen *et al.* und Chen und Nahrstedt [CNS00; CN99] näher betrachtet.

Die Dynamik eines ad hoc Netzes stellt hohe Anforderungen an das Routing. Da Kommunikationsverbindungen unterbrochen werden und auch neue entstehen, ist es bereits schwierig generell einen Pfad zum Zielrechner zu finden, geschweige denn den besten.

Die Routing-Verfahren werden in *proaktives* und *reaktives* Routingverfahren eingeteilt. Die Mischform aus beiden wird mit *hybridem* Routing bezeichnet.

3.1.3.1 Proaktives Routing

Bei den proaktiven Routing-Verfahren wird die benötigte Routinginformation beschafft, bevor eine Nachricht gesendet wird. Hier besteht jedoch ein hoher Signalisierungsbedarf, weil permanent (d. h. in periodischen Zeitintervallen) überprüft werden muss, ob die Informationen noch die notwendige Gültigkeit aufweisen. Gespeichert werden diese Daten in einer Routing Tabelle. Ein Beispiel für diese Kategorie ist das Destination-Sequenced Distance-Vector Routing Protokoll (DSDV) [PB94]. Dieses Verfahren findet Pfade mit minimalem Hop Count, unterstützt jedoch Mobilität nur mäßig, da Routingtabellen i. A. ständig veraltet sind und neu aufgebaut werden müssten.

3.1.3.2 Reaktives Routing

Reaktive Routing-Verfahren erzeugen die Routinginformation nur bei Bedarf, d. h. wenn der Empfänger bekannt ist und eine Übertragung erwünscht wird. Dabei entsteht eine Zeitverzögerung beim Ausliefern der Pakete, da die Information vorher beschafft werden muss. Zu dieser Klasse gehört das Ad hoc On-Demand Distance-Vector Protokoll (AODV), welches eine Erweiterung des DSDV darstellt [PBRD03]. Soll eine Route bestimmt werden, wird eine entsprechende Nachricht an alle Netzteilnehmer gesendet (flooding Broadcast). Ist das Zielgerät erreicht, schickt dieses eine entsprechende Antwort zurück, wobei die Route bestimmt wird. AODV eignet sich gut für mobile ad hoc Netzwerke, sofern die Geschwindigkeit der Benutzer nicht über 5 $\frac{m}{s}$ ansteigt [PR00].

3.1.3.3 Hybrides Routing

Hybrides Routing ist eine Mischform aus reaktivem und proaktivem Routing wie beispielsweise das *Zone Routing* Protokoll [SKD00]. Hier wird die Geräteumgebung in zwei Zonen aufgeteilt. Die Intra-Zone ist die unmittelbare Umgebung eines Gerätes, die sich bis zu drei Hops erstreckt. In diesem Bereich wird proaktives Routing verwendet. Alle anderen Geräte gehören der Inter-Zone an, in der reaktives Routing angewandt wird. Auf diese Weise können die Vorteile beider Verfahren genutzt werden.

3.1.3.4 Andere Routingmechanismen

Multicast Routing (z. B. ODMRP [LSG02]) bietet die Möglichkeiten Daten gleichzeitig an mehrere Knoten zu senden, womit Bandbreite eingespart werden kann. Dieses Verfahren muss von den Netzwerkroutern unterstützt werden. Bei einem Multicast werden mehrere Knoten zu einer Gruppe zusammengefasst, welcher eine Adresse zugewiesen wird. Wird eine Nachricht an diese Adresse gesendet, so empfängt diese jeder Teilnehmer der Gruppe. Die Verwendung von Multicasts in Routingalgorithmen soll die Netzwerklast verringern.

In großen Netzen wird of *hierarchisches* Routing angewandt. Bei diesem Verfahren wird ein großes Netzwerk in mehrere Ebenen unterteilt [Tan97]. Ein Knoten auf einer höheren Ebene ist für ein komplettes Teilnetz der darunter liegenden Ebene zuständig. Ein Beispiel für hierarchisches Routing ist das Internet.

Neuere Verfahren versuchen zunehmend das Routing energieeffizienter zu gestalten, da es im mobilen Umfeld ein Energie-Dilemma darstellt! Zum einen ist auf Grund der Mobilität mehr Signalisierung notwendig, da überprüft werden muss, ob bestehende Routen

noch gültig sind. Zum anderen soll aber so wenig wie möglich gesendet werden, da dieser Vorgang einer der Hauptverbraucher an Energie ist.

Abschließend soll noch angemerkt werden, dass das universelle Routing-Verfahren für ad hoc Netzwerke nicht existiert. Welche Routing-Variante in einem konkreten ad hoc Netz verwendet wird, hängt wesentlich von der Größe des Netzwerks und dem Bewegungseigenschaften der Teilnehmer ab. Ein pauschale Aussage ist nicht möglich und um das richtige Verfahren zu finden, sind intensive Untersuchungen unumgänglich.

3.1.4 Drahtlose Funktechnologien

Bei mobilen ad hoc Netzwerken ist für die Kommunikation eine drahtlose Funktechnologie nötig. Aus diesem Grund werden kurz die Technologien *Wireless LAN* und *Bluetooth* vorgestellt. Beide Technologien finden Einsatz bei der drahtlosen Verbindung zwischen Desktop- und Laptop-Computern und PDAs. Bluetooth findet auf Grund des geringeren Energieverbrauchs auch Einsatz in Mobiltelefonen.

3.1.4.1 Wireless LAN

Unter einem *Wireless LAN* wird die Gruppe der 802.11-Standards des IEEE[2] verstanden. Die Gruppe umfasst die Standards 802.11a - 802.11i, wobei die wichtigsten in Tabelle 3.1 zusammengefasst sind. Die Funkreichweite eines WLANs erstreckt sich innerhalb

Typ	Datenübertragungsrate	Sendeleistung	Frequenzbereich	Kompatibilität zu
802.11	2 Mbit/s	100 mW	2,4 GHz	802.11b
802.11a	6-54 Mbit/s	30 mW	5,0 GHz	-
802.11b	5.5, 11 Mbit/s	100 mW	2,4 GHz	802.11b+/g
802.11b+	22 Mbit/s	100 mW	2,4 GHz	802.11b/g
802.11g	6-54 Mbit/s	100 mW	2,4 GHz	802.11b/b+

Tabelle 3.1: 802.11 IEEE Standards im Vergleich [Gie03]

von Gebäuden auf 10-50 Meter und beträgt außerhalb davon je nach Sichtverbindung 30-550 Meter.

Mit Technologien des 802.11 Standards ist es möglich sowohl infrastrukturbehaftete als auch infastrukturlose ad hoc Netzwerke zu realisieren.

3.1.4.2 Bluetooth

Bluetooth ist ein offener Standard zur drahtlosen Kommunikation. Bezüglich der Funkreichweite existieren drei Leistungsklassen, Class 3 bis Class 1, die von einer Mindestreichweite von 10 Metern bis 100 Meter ausgehen, wenn die Sendeleistung bis auf 100mW erhöht wird [BT03]. Das Bluetooth-Protokoll unterstützt einen asymmetrischen Datenkanal mit

[2]Institute of Electrical and Electronics Engineers

Datenraten von maximal 732,2 kbit/s in eine Richtung und 57,6 kbit/s in die Gegenrichtung oder eine symmetrische Datenverbindung mit 433,9 kbit/s in beide Richtungen. Ab der Version 1.2 erhöht sich die maximale Bandbreite auf bis zu 2,2 Mbit/s.

Das Bluetooth-System stellt sowohl Point-to-Point als auch Point-to-Multipoint-Verbindungen her. Ein Piconet besteht aus einer Ansammlung von zwei bis acht Geräten. Die Bluetooth-Geräte in einem Piconet sind sich ebenbürtig und haben eine identische Implementierung. Dennoch muss ein Gerät als Master und die anderen als Slaves fungieren, wobei der Master alle anderen Teilnehmer im selben Piconet synchronisiert, d. h. er steuert, welches Gerät zu welchem Zeitpunkt sendet. Bluetooth-Geräte können durch Zeitmultiplexverfahren mehreren Piconets angehören, wodurch ein so genanntes Scatternet gebildet wird. Es können bis zu zehn Piconets zu einem Scatternet zusammengeschlossen werden.

3.1.5 Formalisierung eines ad hoc Netzwerks

Netzwerke und somit auch ad hoc Netzwerke werden i. A. als Graph $G = (V, E)$ modelliert, z. B. Diestel [Die00], wobei V die Knotenmenge und E die Kantenmenge repräsentiert. Hierbei wird jeder (mobile) Host als Knoten $v \in V$ dargestellt. Können zwei Geräte - die durch die Knoten v_i und v_j in G repräsentiert werden - direkt miteinander kommunizieren, so sind sie in G mit einer Kante $e = (v_i, v_j)$ verbunden. Ein Beispiel hierfür ist in Abbildung 3.3a zu sehen.

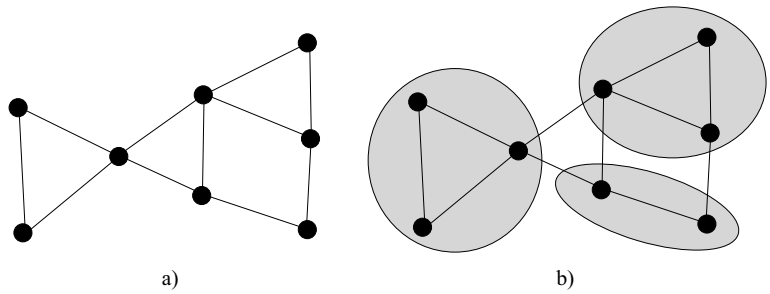


Abbildung 3.3: Graph eines Netzwerkes (a) und eine mögliche Aufteilung in drei Segmente (b)

In einem ad hoc Netzwerk können zwei Geräte A und B, die die Knoten v_i und v_j repräsentieren, direkt miteinander kommunizieren, wenn ihre Entfernung $d(A, B)$ zueinander kleiner als ihr jeweiliger Kommunikationsradius r_A bzw. r_B ist, d. h. es muss $d(A, B) < \min(r_A, r_B)$ gelten.

In einem ad hoc Netzwerk sind zudem die Geräte mobil, so dass sich die Entfernung von Geräten ändern kann. Für den Graphen G bedeutet dies, dass er sich mit der Zeit ändert und aus $G = (V, E)$ eine zeitabhängige Funktion $G(t) = G(V(t), E(t))$ wird. Somit kann Abbildung 3.3 nur einen Schnappschuss eines ad hoc Netzwerkes darstellen.

Nachdem definiert wurde, wie ein Netzwerk modelliert wird, kann erläutert werden, was unter Segmentierung eines Netzwerks zu verstehen ist.

Definition 1: *Segment*
Ein Segment $S = (V_S, E_S)$ ist ein Teilgraph eines Graphen $G = (V, E)$, d. h. $V_S \subseteq V$ und $E_S \subseteq E$, wobei sämtliche $v \in V_S$ zusammenhängen, d. h. für zwei Knoten $v_i \in V_S$ und $v_j \in V_S$ existiert ein Pfad $P = \{e_1 = (v_i, v_{i+1}), e_2 = (v_{i+1}, v_{i+2}), \ldots, e_m = (v_{j-1}, v_j)\}$, wobei $\forall e_i \in P : e_i \in E_S$ gilt und die v_i paarweise verschieden sind.
Da ein Graph aus mehreren Segmenten bestehen kann, sei mit $\mathcal{S} = \{S_1, S_2, \ldots S_n\}$ die Menge der Segmente bezeichnet.

□

Die *Segmentierung* eines Netzwerkes entspricht der Aufgabe, einen Graphen $G = (V, E)$ in mehrere Segmente S_i zu zerlegen, wobei jeder Knoten v der Knotenmenge V des Graphen G in mindestens einem Segment S_i vorkommen muss, d. h. $\forall v_j \in V, \exists S_i$ mit $v_j \in S_i$. Ein Beispiel ist in Abbildung 3.3b dargestellt.

Gilt für je zwei Segmente S_i und S_j, dass $S_i \cap S_j = \emptyset$, wobei $i \neq j$, und $\bigcup S_i = V$, so liegt eine *Partition* von G vor.

Durch die Segmentierung wird ein großes ad hoc Netzwerk in mehrere kleine Teilnetze zerlegt. Hierdurch wird erreicht, dass die Teilnehmeranzahl reduziert wird, wenn sich die Gruppenbildung auf ein Teilnetz beschränkt. Diese Reduzierung ist notwendig, da in zu großen ad hoc Netzen eine Gruppenbildung nicht möglich ist, weil eine zu große Anzahl an Geräten ausfallen kann und außerdem eine Vielzahl an Nachrichten versendet werden muss, um eine Gruppe zu erzeugen.

3.1.6 Eigenschaften von ad hoc Netzwerken

Im Nachfolgenden werden ad hoc Netzwerke mathematisch untersucht. Grundlage ist hierfür die probabilistische Graphentheorie. Ad hoc Netzwerke können durch probabilistische Graphen modelliert werden, da die Struktur eines ad hoc Netzwerkes nicht festgelegt ist, sondern sich zufällig ergibt. Aus diesem Grund können Ergebnisse der probabilistischen Graphentheorie auf ad hoc Netzwerke übertragen werden.

Die probabilistische Graphentheorie versucht Aussagen zu formulieren, die von *fast allen*[3] Graphen erfüllt werden. Zudem wird untersucht, wie sich bestimmte Eigenschaften, z. B. der Zusammenhang eines Graphen, mit zunehmender Konnektivität (Kanten - Knoten - Verhältnis) ändert.

3.1.6.1 Mathematische Modelle

In der probabilistischen Graphentheorie [Bol85; Pal85] sind zwei Wahrscheinlichkeitsmodelle für Graphen vorherrschend.

Im ersten Modell (Model A) existiert eine Kante in einem Graphen G mit einer Wahrscheinlichkeit p. Die Wahrscheinlichkeit, dass ein Graph mit n unterscheidbaren Knoten und q Kanten zufällig zu Stande kommt, beläuft sich u. a. nach Bollobás [Bol85] auf

$$P(G) = p^q (1-p)^{\binom{n}{2}-q}. \tag{3.1}$$

[3] Formal bedeutet dies, dass

$$\lim_{n\to\infty} P(G \text{ erfüllt Aussage } A) = 1,$$

d. h. die Wahrscheinlichkeit, dass ein Graph G eine Eigenschaft erfüllt, geht für eine große Anzahl untersuchter Graphen gegen eins.

Modell B geht davon aus, dass unter allen möglichen Graphen mit n unterscheidbaren Knoten und q Kanten jeder Graph mit der gleichen Wahrscheinlichkeit auftritt. Da in einem Graphen mit n Knoten maximal $\binom{n}{2}$ Verbindungen existieren können, aber nur q Kanten vorhanden sind, ist die Wahrscheinlichkeit durch

$$P(G) = \frac{1}{\binom{\binom{n}{2}}{q}} \tag{3.2}$$

gegeben.

Zwischen der Kantenanzahl q und der gegebenen Wahrscheinlichkeit p existiert der Zusammenhang

$$q = \left\lfloor p\binom{n}{2} \right\rfloor, \tag{3.3}$$

wobei die Klammern $\lfloor\rfloor$ die Abrundung bezeichnet.

3.1.6.2 Grapheigenschaften

Nun werden kurz die Eigenschaften von Graphen angegeben, die sich mit zunehmender Kantenwahrscheinlichkeit p ergeben.

Nach Diestel [Die00] weist fast jeder Graph bei einer Kantenwahrscheinlichkeit p unterhalb von $\frac{1}{n^2}$ nur isolierte Kanten auf. Nimmt p auf $\frac{1}{n^{3/2}}$ zu, treten in fast jedem Graph die ersten Kanten zwischen einzelnen Knoten auf, die bis zu einer Kantenwahrscheinlichkeit von $\frac{1}{n}$ zu Bäumen anwachsen. Erst ab dieser Wahrscheinlichkeit tauchen die ersten Kreise in fast allen Graphen auf. Erhöht sich die Wahrscheinlichkeit ein wenig auf $\frac{\ln n}{n}$, sind fast alle Graphen zusammenhängend. Wird p nun nur marginal auf $(1+\epsilon)\frac{\ln n}{n}$ erhöht, hat fast jeder Graph einen Hamiltonskreis [Die00].

Somit lässt sich erkennen, dass sich in einem Zufallsgraphen der Zustand oder seine Eigenschaften nicht allmählich, sondern sehr abrupt ändern. Die probabilistische Graphentheorie formuliert dieses Phänomen mit Hilfe von Schwellwertfunktionen $t(c, n)$. Übersteigt der Wert von c einen gewissen Schwellwert, so besitzen fast alle Graphen eine bestimmte Eigenschaft, wogegen im anderen Fall die Eigenschaft von fast keinem Graphen erfüllt wird [Pal85].

Als Beispiel ist die Existenz von isolierten Knoten, d. h. Knoten, die mit keinem anderen Knoten im Graph verbunden sind durch die Schwellwertfunktion

$$t(c, n) = \frac{1}{2}cn \ln n \tag{3.4}$$

gegeben. Für $c > 1$ existiert in fast jedem Zufallsgraphen kein isolierter Knoten. Im anderen Fall, $0 < c < 1$, existieren in fast jedem Zufallsgraphen isolierte Knoten. Für den Fall $c = 1$ kann keine allgemeine Aussage getroffen werden und es müssten weitere Verfahren angewandt werden.

Ellis *et al.* [EJY04] übertragen dieses Ergebnis auf ad hoc Netzwerke, in denen Knoten nur miteinander durch Kanten verbunden sind, wenn eine maximale Distanz nicht überschritten ist. Die Ergebnisse beziehen sich auf eine Einheitsfläche, d. h. Flächengröße einer Längeneinheit.

$$t(c,n) = c \cdot \sqrt{\frac{\ln n}{n}} \tag{3.5}$$

Für $c > 1$ ist die Anzahl X an isolierten Knoten im zufälligen ad hoc Graphen gleich null. Für $0 < c < 1$ gilt $X \sim n^{1-c^2}$.

Als nächstes wird betrachtet, wie der Zusammenhang eines Graphen von der Kantenwahrscheinlichkeit p abhängt. Bollobás [Bol85] gibt für die Wahrscheinlichkeit P_n, dass ein Graph mit n Knoten und Kantenwahrscheinlichkeit p zusammenhängend ist,

$$P_n = 1 - \sum_{k=1}^{n-1} \binom{n-1}{k-1} p(1-p)^{k(n-k)} \tag{3.6}$$

als rekursive Formel an. Hiermit wird bei der Analyse einer MoPiDiG Anwendung die Anzahl an möglichen Teilnehmern nach oben abgeschätzt.

3.1.7 Segmentierung von Netzwerken

Im Nachfolgenden sollen einige Methoden vorgestellt werden, wie ein Netzwerk in Segmente aufgeteilt werden kann. Es existieren Methoden, die disjunkte Segmente generieren und Verfahren, bei welchen Knoten mehreren Segmenten angehören können[4]. Ferner ist die Größe eines Segments noch ein Designkriterium, in welchem sich die Verfahren unterscheiden.

3.1.7.1 Annahmen

Im Folgenden wird angenommen, dass jedes Gerät seine direkten Nachbarn „kennt“. Die direkten Nachbarn eines Gerätes sind alle anderen Geräte, die sich innerhalb seines Kommunikationsradius befinden.

Die direkten Nachbarn N_{v_i} eines Knoten v_i ist eine Teilmenge der Knotenmengen V mit $N_{v_i} = \{v_j \in V | (v_i, v_j) \in E\}$ $(i \neq j)$. Zusätzlich ist jeder Knoten mit einer global eindeutigen ID versehen.

Beide Annahmen sind i. A. auch in der realen Welt erfüllt und stellen keine Einschränkungen dar. Zum einem ist in einem Netzwerk jeder Teilnehmer mit einem eindeutigen Bezeichner (z. B. Email-Adresse) assoziiert, der als ID benützt werden kann. Zum anderen kennt im ad hoc Netzwerk ein Gerät i. A. seine direkten Nachbarn, da dies für eine Kommunikation unabdingbar ist.

Im Weiteren wird vorausgesetzt, dass sämtliche Geräte in G den gleichen Kommunikationsradius r besitzen. Diese Annahme ist in der realen Welt kaum erfüllt, erleichtert aber die Modellierung des Problems. Durch diese Annahme kann die Kommunikation als symmetrisch angesehen werden, d. h. sendet ein Knoten A eine Nachricht an den Knoten B, so kann auch B an A eine Nachricht senden.

[4] In diesem Abschnitt und für den Rest der Ausführungen wird der Prozess, in dem ein Kommunikationsnetzwerk in Subnetze aufgeteilt wird, als Segmentierung bezeichnet. Die einzelnen Subnetze werden Segmente benannt. In der englischsprachigen Literatur wird der Vorgang als „Clustering“ und werden die Subnetze als „Cluster“ bezeichnet. Die verwendete Nomenklatur wurde gewählt um Verwechslungen auszuschließen, da der Terminus „Clustering“ im Zusammenhang mit der Gruppierung von Daten verwendet wird.

3.1.7.2 Algorithmen

Gerla und Tsai [GT95] stellen zwei Segmentierungsalgorithmen für ad hoc Netwerke vor, *lowest-ID Segmentierung* und den *highest-connectivity* Algorithmus. Beiden Algorithmen ist gemein, dass in einem Segment ein so genannter *Head* existiert. Dieser Head kann als Koordinator des Segments angesehen werden. Ist ein Knoten kein Head, wird dieser Knoten einem Head zugeordnet. Außerdem sind zwei Heads nie direkte Nachbarn und die entstehenden Segmente sind nicht notwendigerweise disjunkt, d. h. es gibt Knoten, die in zwei Segmenten gleichzeitig vorkommen. Diese Knoten werden als Gateway-Knoten bezeichnet. Die zwei Algorithmen differieren nur in der Art und Weise, welcher Knoten zum Head wird.

Lowest-ID Segmentierung: Jeder Knoten besitzt eine im Gesamtnetz eindeutige ID. Diese wird mit jedem Broadcast verschickt, um sie der Nachbarschaft bekannt zu machen. Empfängt ein Knoten nur höhere IDs als seine eigene, wird er automatisch zum Head. Bekommt ein Knoten auch niedrigere IDs, so wird derjenige zu seinem Head, der ihm die niedrigste ID zugesandt hat. Alle dem Head angrenzenden Knoten gehören zu einem Segment. In Abbildung 3.4a sind die Knoten 1, 2 und 4 jeweils Head. Die Knoten 8 und

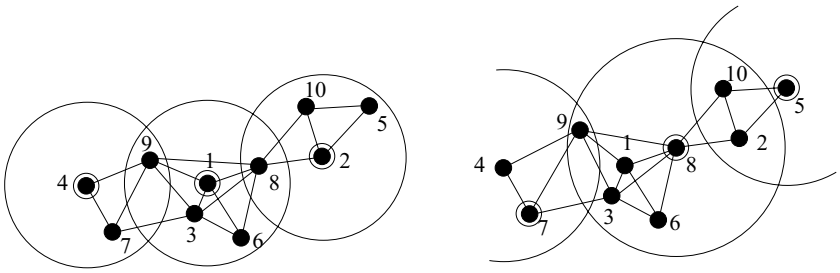


Abbildung 3.4: Lowest-ID Segmentierung (a) und Highest-connectivity Algorithmus (b). Umrandete Knoten stellen Heads dar.

9 sind in zwei Segmenten vorhanden und können somit als Gateway fungieren (siehe auch Abbildung 3.4a).

Highest-Connectivity Segmentierung: Bei dieser Variante wird derjenige Knoten Head, der am meisten direkte Nachbarn aufweisen kann. Im Gegensatz zur ID wird hier die Anzahl an Nachbarn nach außen gegeben. Im Fall eines Gleichstandes an Verbindungen muss ein zweiter Parameter hinzugezogen werden (evtl. Zufallszahl oder ID). Ein Beispiel ist in Abbildung 3.4b zu sehen. In dieser Abbildung ist ebenfalls ersichtlich, dass beide Algorithmen unterschiedliche Segmente erzeugen. Im zweiten Fall sind die Knoten 4, 5 und 8 Head, obwohl es sich um den gleichen Graphen wie im ersten Algorithmus handelt.

Der Head eines Segments ist in beiden Fällen für das Aufrechterhalten des Segments verantwortlich. Kommen neue Knoten hinzu oder treten welche ab, kann dies zu einem Wechsel des Heads führen. Laut Gerla und Tsai [GT95] treten bei dem *Lowest-ID Segmentierungs Algorithmus* weniger Änderungen der Heads auf als beim *Highest-Connectivity*

Segmentierungs Algorithmus, was die Autoren mit Simulationsergebnissen belegen.

Die Arbeiten von Gerla und Tsai sowie Basagni [GT95; Bas99] liefern disjunkte Segmente und sind nicht hierarchisch organisiert. Banerjee und Khuller [BK00] zeigen einen konträren Ansatz. Die gebildeten Segmente sind nicht disjunkt und können eine Hierarchie (Knoten - Segment - Supersegment) bilden.

Ein Segment ist bei Banerjee und Khuller [BK00] *„a subset of vertices whose induced graph is connected“*. Zwei Segmente können Knoten gemeinsam haben und die Segmentgröße soll nach oben und unten beschränkt sein und variiert zwischen k und $2k$. Die Kommunikationsreichweite ist unter allen Knoten gleich r_K. Der Algorithmus zielt darauf

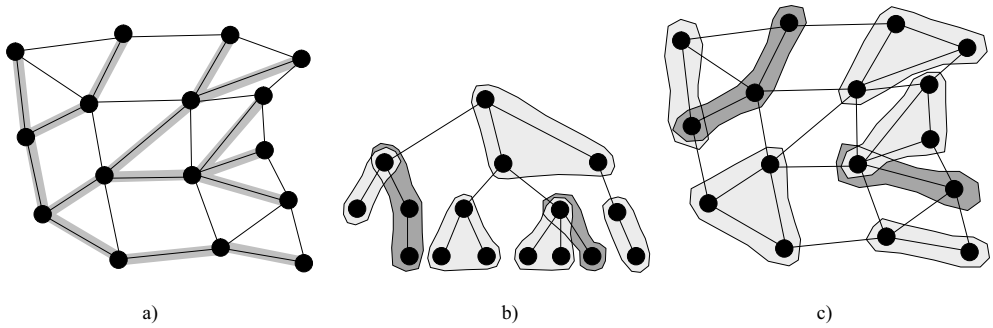


Abbildung 3.5: a) zeigt das Ausgangsnetzwerk mit grau unterlegtem Spannbaum. In b) ist der Spannbaum mit Segmenten nach Banerjee und Khuller [BK00] zu sehen. Die Segmente im Netzwerk zeigt ebenfalls b)

ab, in einem Spannbaum des Graphen G nach Teilgraphen zu suchen, wie in Abbildung 3.5 zu sehen ist.

Ein Baum $T \subseteq G$ heißt *Spannbaum* von G, wenn er G aufspannt, d. h. wenn $V(T) = V(G)$ ist. Laut Graphentheorie existiert in jedem zusammenhängenden Graph ein Spannbaum [Die00].

Ferner sei v ein Knoten von G, und $T(v)$ ein Teilbaum von G mit v als Wurzel. Die Größe, d. h. die Anzahl an Knoten von $T(v)$, sei mit $|T(v)|$ bezeichnet. $C(v)$ sei die Menge aller Kinderknoten von v in T. Zuerst wird ein Knoten u gesucht mit $|T(u)| \geq k$ mit der Einschränkung, dass für jedes $v \in C(u) : |T(v)| < k$. Die Segmentierung wird durch die einzelnen Teilbäume erreicht. Bestehe $C(u)$ aus ℓ Knoten $v_1, \ldots, v_\ell$. Es werden die $T(v_i)$ betrachtet und jeweils so viele Zweige zu einem Segment zusammengefügt, bis die Minimalanzahl erreicht ist. Das Hinzufügen eines Teilbaumes kann das Maximalkriterium nicht sprengen, da alle Subbäume nur $k-1$ Knoten aufweisen. Der Knoten u wird immer zum Segment hinzugefügt, damit die Knoten auch miteinander verbunden sind. Ist ein Segment gebildet, so werden die darin enthaltenen Knoten aus dem Baum gelöscht und es beginnt wieder mit dem Schritt der Teilbaumsuche.

Problem dieses Algorithmus ist jedoch, dass ein beliebiger Knoten u theoretisch zu allen Segmenten gehören kann. Hierfür wird jedoch eine Lösung von Banerjee und Khuller [BK00] diskutiert. Durch die Verwendung von Bäumen arbeitet dieses Verfahren mit linearer Komplexität und kann deshalb auch für große Graphen angewandt werden.

Wenn sich neue Knoten dem Segment anschließen, so ist das bis zu einem oberen Schwellwert möglich. Ist dieser überschritten, so teilt sich das Segment in zwei Segmente auf. Verlassen hingegen Knoten das Segment und der untere Schwellwert wird unterschritten, so versuchen die restlichen Knoten sich anderen Segmenten anzuschließen.

Krishna *et al.* [KVCP97] thematisieren ebenfalls das Problem der Segmentierung. In diesem Fall wird die Segmentierung für das Routing verwendet. Aus diesem Grund müssen sich die Segmente auch überlappen, um eine Route zwischen entfernten Segmenten zu finden. Knoten, die zu mehreren Segmenten gehören, nennt man *Boundary Nodes*.

Der Knoten (das ad hoc Gerät), welcher als Head fungiert, weist auf Grund des zusätzlichen Kommunikationsaufwandes einen erhöhten Energieverbrauch auf, was sich insbesondere bei kleinen mobilen Geräten als nachteilig auswirkt. Deshalb verzichtet das Verfahren auf die Existenz eines Heads. Stattdessen kennt jeder Knoten nicht nur seine Nachbarn, sondern auch alle *Boundary Notes* und Segmente. Bei einer Änderung der Segmentstruktur müssen diese Daten jedoch jeweils in jedem Knoten aktualisiert werden.

Bisherige Algorithmen bezogen sich nur auf Segmente, deren Knoten maximal nur einen *Hop* von einem Head entfernt sind. Chen und Stojmenovic [CS99] erweitern dies auf k-hop-Segmentierung. Kennt bei Gerla und Tsai [GT95] ein Knoten nur seine nächsten Nachbarn, sind bei dem Verfahren *k-CONID* von Chen und Stojmenovic [CS99] alle Nachbarn bekannt, die sich innerhalb k Hops befinden.

Für $k > 2$ ist dieses Verfahren gemäß Aussagen der Autoren jedoch nicht mehr sinnvoll anwendbar, weil die Segmente in diesen Fällen sehr groß sind, d. h. die Anzahl an Geräten übersteigt eine Art *kritische Masse* und es treten zu viele Strukturänderungen auf, so dass von der Existenz eines Segments zu keiner Zeit gesprochen werden kann.

Ein weiterer Bestandteil ist die Kollisionsfreiheit des Algorithmus. Bei der Segmenterzeugung können mehrere Knoten gleichzeitig neue Segmente bilden wollen. Dies führt zum quasi-gleichzeitigen Senden von Nachrichten und somit zu Kollisionen. Zur Lösung wird der Graph in einen Spannbaum transformiert und dieser gemäß der Depth-First-Suche traversiert. Die Traversierung macht sich durch ein Umherwandern eines *Tokens* bemerkbar. Nur der Knoten, der momentan das Token besitzt, kann Änderungen in der Segmentstruktur initiieren.

Die Segmentierung kann auf mehreren Ebenen erfolgen. Segmentierung auf einer Ebene (Level-1 Segmentierung) skaliert [CS99] bis auf einige hundert Knoten und ist deshalb für unsere Ausführung mehr als ausreichend. Messungen mit zwei Compaq IPaQs ergeben, dass in freier Natur der Kommunikationsradius mit WLAN ungefähr 300 Meter beträgt. Innerhalb von Gebäuden reduziert sich der Kommunikationsradius auf maximal 60 Meter (auf Reflexionen, Streuungen etc. zurückzuführen), sofern keine Wand zwischen den Geräten liegt. Diese Messungen ergeben, dass es für eine Segmentierung primär nicht notwendig ist, mehrere Hops zu berücksichtigen. Die 60 Meter reichen in Gebäuden für die meisten Meeting- und Büroräume leicht aus. Was noch nicht untersucht wurde, ist, ob der Kommunikationsradius auch von der Geschwindigkeit des Sendegerätes abhängt. In anderen Fällen könnten bis zu n Hierarchien gebildet werden (Level-n Segmentierung).

Badrinath *et al.* [BAI94] beschreiben die Probleme mit Algorithmen in mobilen Umgebungen. Ferner legen sie ein Konzept vor, wie Algorithmen im dynamischen Umfeld

aufgebaut sein sollen. So trennen die Autoren die eigentlichen Algorithmen von der Mobilität, d. h. unter einer Algorithmusschicht existiert eine untere Ebene, die verantwortlich ist zu kontrollieren, ob Kommunikationspartner noch vorhanden sind, bzw. ob Nachrichten von einer Gegenseite empfangen wurden. Doch konzentrieren sich die Autoren primär auf ad hoc Umgebungen mit Infrastruktur (Basisstation), wie sie in Abschnitt 3.1.1 beschrieben wurden. Die vorgestellte Zwei-Schichten-Architektur kann jedoch mit einigen Modifikationen auch für infrastrukturlose ad hoc Netzwerke eingesetzt werden.

Galluccio *et al.* [GMP02] fassen Geräte zu Segmenten zusammen, die eine gewisse Entfernung nicht überschreiten. Die Segmentierung basiert somit auf dem Abstand der Geräte zueinander. Das Framework *MANGO* verfügt über eine hierarchische Architektur um die Routingproblematik zu lösen.

Jedes Segment hat einen *Group Leader* (GL). Dieser ist verantwortlich für die Aktualisierung der Ortsinformation und muss die Segmentmitglieder über Änderungen informieren, da die Entfernung zwischen GL und restlichen Teilnehmern einen Schwellwert nicht überschreiten darf. Der GL verbraucht mehr Energie, deshalb soll die Rolle des GL innerhalb des Segments rotieren, damit ein gewisser Grad an Fairness erzielt wird. Dies stellt somit einen Unterschied zu Meissner und Musunoori sowie Roman und Hazemi [MM03; RHH01] dar. Durch die Rotation des GL wird allerdings die Problematik hinsichtlich seiner Ausfallsicherheit nicht gelöst.

Dass Mobilität jedoch nicht notwendigerweise eine negative Eigenschaft sein muss, zeigen Gerla *et al.* [GXH03]. Die Autoren fassen Geräte zu Gruppen zusammen, die ähnliche Geschwindigkeitsvektoren besitzen. Beispiele hierfür sind Kraftfahrzeuge auf Autobahnen, die sich mit gleicher Geschwindigkeit fortbewegen. Gerla *et al.* wenden diese Gruppen für das Routing in ad hoc Netzwerken an, da die so entstehenden Teilnetze bei minimaler Geschwindigkeitsdifferenz hinsichtlich ihrer Bewegung nur quasi-statisch sind.

In Tabelle 3.2 werden sämtliche vorgestellte Segmentierungsverfahren hinsichtlich aus-

	disjunkte Segmente	statischer Gruppenführer	wechselnde Gruppenführer	k-hop Kommunikation	Segment-hierarchie möglich	Segmentierungs-grundlage
Gerla und Tsai [GT95]	★	★				r_K
Basagni [Bas99]	★	★				r_K
Banerjee und Khuller [BK00]					★	r_K
Krishna *et al.* [KVCP97]						r_K
Chen und Stojmenovic [CS99]		★		★		r_K
Galluccio *et al.* [GMP02]			★			r_K
Gerla *et al.* [GXH03]		★				v

Tabelle 3.2: Vergleich existierender Lösungen für die Segmentierung auf Netzwerkebene

gewählter Kriterien miteinander verglichen.

Die erste Spalte gibt an, ob die erhaltenen Segmente disjunkt sind oder nicht. Spalte

zwei und drei beschreiben, ob ein Gruppenführer (head) existiert und ob dieser sich ändert oder nicht. Die nächste Spalte gibt an, ob innerhalb der Segmente eine k-Hop Kommunikation möglich ist oder ob sich die Kommunikation auf einen Hop beschränkt. Die fünfte Spalte gibt an, ob mit dem Verfahren eine Hierarchie möglich ist. In der letzten Spalte (Segmentierungsgrundlage) wird noch erläutert, welche Knoten zu Segmenten zusammengefügt werden. Bis auf den letzten Eintrag werden ausnahmslos Knoten zu Segmenten vereinigt, die sich innerhalb einer oder mehrerer Kommunikationsradien (r_K) befinden. Beim letzten Verfahren [GXH03] werden Knoten mit ähnlicher Geschwindigkeit v zu Segmenten zusammengefügt.

3.2 Profile

Die Gruppenbildung in einer MoPiDiG Anwendung erfolgt anhand von Benutzerprofilen. Aus diesem Grund werden Profile in diesem Abschnitt näher betrachtet. Nach einer Begriffsklärung wird auf die Darstellungsmöglichkeiten von Profilen eingegangen. Anschließend werden manuelle und automatische Methoden vorgestellt, wie Benutzerprofile erstellt werden können. Dieser Abschnitt schließt mit einer kurzen Betrachtung von Sicherheitsaspekten in Verbindung mit Profilen.

3.2.1 Begriffsklärung

Der Profilbegriff wird in der Literatur stark diskutiert, wobei eine allgemeingültige Definition nicht vorliegt. Die verschiedenen Auffassungen unterscheiden sich jedoch nur geringfügig.

Kobsa [Kob93] definiert ein Profil als das Wissen, die Pläne und Präferenzen einer Person. Unter einem Profil wird somit eine umfassende Datensammlung zu einem Objekt verstanden. Diese Sammlung kann als Menge von Eigenschaften oder Attributen mit dazugehörigen Werten angesehen werden. Ein Benutzerprofil kann beispielsweise beschreiben, welche Eigenschaften ein Benutzer bevorzugt oder welche Arbeiten er ausführen kann.

Pohl [Poh96] stellt den Profilbegriff dem des Benutzermodells gleich. Unter diesem versteht der Autor Informationen über das Wissen, die Ziele, die Präferenzen, etc. einer Person.

Adomavicius und Tuzhilin [AT01] unterscheiden im Datenmodell zwischen *sachbezogenen* und *transaktionellen* Daten. Sachbezogene Daten beschreiben, wer die Person ist (Name, Adresse, Geburtsdatum, etc.), wogegen die transaktionellen Daten die Aktionen einer Person mit Regeln beschreiben (Kaufdaten von Produkten, Einkaufsort, Kaufpreis etc.). Aus den transaktionellen Daten können ferner neue sachbezogene Daten abgeleitet werden.

Obige Profildefinitionen sind informal, weswegen im Folgenden eine formale Definition eines Profils angegeben wird.

Definition 2: *Profil*
Sei $\Pi = \Pi_1 \times \ldots \times \Pi_m$ ein Profilraum mit endlicher Dimension m, wobei die $\Pi_i \in \Pi_i$ einem beliebigen Datentyp entsprechen. Ein Punkt $P \in \Pi$ mit $P = (p_1, \ldots, p_m)$ entspricht einem Profil, wobei die p_i als Profileinträge bezeichnet werden.

Der Wertebereich der p_i ist a priori nicht festgelegt, z. B. $p_1 \in \mathbb{R}$, $p_2 \in [0;1]_{\mathbb{R}}$, $p_3 \in \{\text{Rot, Grün, Blau}\}$, $p_4 \in \mathbb{N}$.

□

Als Beispielprofil soll hier ein Profil für das „Public Transport on Demand“-Szenario aus Abschnitt 1.1.2.1 betrachtet werden. Als Eintrag ist die Zielposition sowie Verkehrsmittel für dieses Beispiel ausreichend. Die Zielposition wird als Punkt im zweidimensionalem Koordinatensystem angegeben, so dass hierfür zwei Profileinträge notwendig sind. Somit ist $\Pi = \Pi_1 \times \Pi_2 \times \Pi_3 = \text{XPosition} \times \text{YPosition} \times \text{Verkehrsmittel}$.

Die zwei Werte für die Zielposition können durch reelle Werte aus $\mathbb{R}$ angegeben werden. Das Verkehrsmittel wird durch einen Aufzählungstyp bestimmt der aus den Werten Bus, Taxi, U-Bahn besteht. Ein Punkt im Profilraum könnte somit die Gestalt (100, 100, Bus) haben, was bedeutet, dass eine Person zur Position (100, 100) mit dem Bus reisen will.

Der Begriff des Profils ist ein spezieller Fall von Kontextinformation, die bereits in Abschnitt 2.3.1 beschrieben wurde. Vereinigt der Kontextbegriff sämtliche persönliche Daten und umgebende Parameter auf sich, so beschränkt sich ein Profil auf die persönlichen Eigenschaften einer Person.

Die obige Definition ist sehr allgemein und impliziert eine Profildarstellung in einem m-dimensionalen Raum. Im nächsten Abschnitt werden weitere Möglichkeiten der Profildarstellung angegeben.

3.2.2 Profildarstellung

Es gibt verschiedene Ansätze zur Modellierung von Benutzerprofilen. Typischerweise werden die Attribute in einem Profil inhaltlich in einzelne Abschnitte gegliedert und hierarchisch aufgebaut. Ausgehend von einer relativ groben Beschreibung werden die einzelnen Punkte schrittweise konkretisiert. Abbildung 3.6 zeigt dies am Beispiel von Freizeitbeschäftigungen.

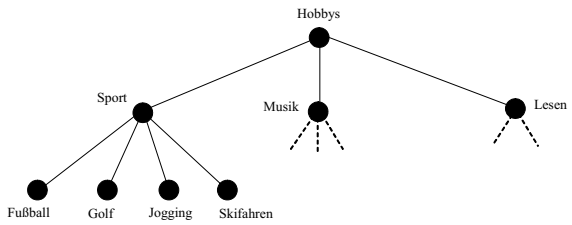


Abbildung 3.6: Hierarchische Konkretisierung von Profilen

3.2.2.1 Key-value pair

Eine einfache und weit verbreitete Methode ein Profil zu definieren sind *Key-value pairs*. Bei dieser Methode wird ein Attribut durch einen *Key* beschrieben. Ein *value* repräsen-

tiert den Eintrag für dieses Attribut für einen konkreten Benutzer. Durch die Einführung rekursiver Strukturen können bei diesem Modell hierarchische Profile erstellt werden.

Für die Repräsentation eines Profils bietet sich eine Darstellung in der Extensible Markup Language (XML) [BPMY04; BPSM+04] an, da damit eine einfache und übersichtliche Strukturierung von Daten erreicht werden kann. Abbildung 3.7 stellt ein Beispielprofil im

```
<NAME>John Smith</NAME>
<WORKINGHOURS>
    <START>0700</START>
    <END>1600</END>
</WORKINGHOURS>
<BREAK>
    <START>0900</START>
    <DURATION>15</DURATION>
</BREAK>
<MACHINEASSOCIATIONS>
    <MACHINE>drill</MACHINE>
    <MACHINE>cutter</MACHINE>
</MACHINEASSOCIATIONS>
```

Abbildung 3.7: Beispielprofil für eine Anwendung im Fertigungsumfeld

Fertigungsumfeld dar. Der Arbeiter kann seine Arbeitszeit und seine Pausen definieren. Ferner sind im Profil noch sämtliche Maschinen aufgeführt, die vom Profilinhaber bedient werden können.

XML ist jedoch nur bedingt geeignet für eine effiziente Speicherung des Profils, insbesondere bei sehr umfangreichen oder dynamischen Daten. Die Markup Sprache kann aber sehr gut zur Veranschaulichung des Konzepts eines Benutzerprofils dienen, da auf diese Weise definierte Profile auch „für den Menschen“ lesbar sind.

Für mobile Endgeräte bietet sich außerdem WAP Binary XML (WBXML) an. Die Alternative WBXML dient dazu, die im XML-Format vorliegenden Informationen in Binärdaten umzuwandeln. WAP Binary XML wurde unter anderem deshalb entwickelt, um die Größe der zu übertragenden XML-Dokumente zu reduzieren.

3.2.2.2 Logikbasiertes Modell

Im logikbasierten Modell wird ein Profil als Konkatenation von *logischen Termen* A_i aufgefasst:

$$A_1 \wedge A_2 \wedge \ldots \wedge A_n$$

Die A_i liegen z. B. in Prolog als Fakten in einem Programm vor. Prolog zeichnet sich durch eine mächtige Repräsentationssprache aus. Außerdem können dem System Regeln hinzugefügt werden und Anfragen an die Datenbank gestellt werden [BBH97].

Anstelle der Prolog Fakten können auch Ausdrücke aus der Prädikatenlogik verwendet werden, welche noch ausdrucksstärker sind und aus diesem Grund von Appelt und Pollak [AP91] diskutiert werden. Die Verwendung von Logiken höherer Ordnung (z. B. Modallogik oder Temporallogik) erlauben zusätzlich die Modellierung von Annahmen.

3.2.3 Kontextmodellierung

Der Begriff des Kontextes wurde bereits in Abschnitt 2.3.1 erläutert. Nun soll noch dessen Modellierung kurz erklärt werden.

Kontextmodellierung ist mit zusätzlichen Herausforderungen konfrontiert. Bei Kontextinformationen kommen zu den Benutzerdaten noch permanent Sensordaten hinzu, die ins Modell mit aufgenommen werden müssen. Aus einer Vielzahl an Sensordaten können mit Hilfe einer Art *Sensorfusion* neue Benutzerdaten eruiert werden. Oft ist es auch notwendig den historischen Kontext zu betrachten, weswegen Mechanismen notwendig sind, die es erlauben Kontextinformationen zeitlich den Benutzerdaten zuzuordnen.

Nigel *et al.* [DCMF99] schlagen ein objektorientiertes Verfahren zur Modellierung vor. Für jede Ortsinformation (z. B. GPS Daten) wird ein Objekt generiert. Sukzessive wird dieses Objekt mit „Leben“ gefüllt. Der Ortskoordinate wird eine reale Adresse und Funktion zugeordnet (z. B. Restaurant). Zusätzliche Informationen (Öffnungszeiten, Speisekarte) konkretisieren und vervollständigen das Objekt. Objekte sind untereinander mit Querverweisen verbunden. Die Attribute in den Benutzerdaten können ebenfalls zu verschiedenen Objekten verweisen.

3.2.4 Erzeugungsmöglichkeiten von Profilen

In diesem Abschnitt wird erklärt, wie Benutzerprofile generiert werden können. Es werden jedoch keine Methoden im Detail erläutert, sondern nur Ansätze und Möglichkeiten aufgezeigt, wie Profile erstellt werden können.

Im MoPiDiG Framework stellen Profile die Rolle der zu gruppierenden Objekte dar, wobei in den folgenden Kapiteln vorausgesetzt wird, dass eine Methode ein Benutzerprofil zu generieren gegeben ist und für das Framework verwendet werden kann.

3.2.4.1 Manuelle Profilerzeugung

Die einfachste Möglichkeit ein Profil zu erstellen, ist die manuelle Generierung durch den Benutzer. Bei dieser Methode kann der zukünftige Profileigentümer mit einer Fülle von Fragen konfrontiert werden, deren Beantwortung das Profil generiert [Yoc96].

Diese Methode ist jedoch sehr zeitaufwändig - die Generierungszeit hängt von der Profilgröße ab - und stellt für den betroffenen Benutzer eine mühsame und unangenehme Tätigkeit dar.

Da Profile nicht statisch sind, sondern sich die Interessen der betreffenden Person mehr oder weniger permanent ändern, ist der Benutzer auch selbst für die Aktualisierung der Profilinformationen verantwortlich. Somit ist der Anwender kontinuierlich mit Arbeiten am Profil konfrontiert, was die Akzeptanz und Verwendung in starkem Maße gefährdet.

Wird ein Profil in exakt definierten Anwendungsbereichen verwendet, ist es möglich, das komplette Profil oder Teile davon dem Benutzer in einem initialisierten Zustand zu übergeben. So kann eine Abteilung in einem Unternehmen jedem Angestellten ein abteilungsspezifisches Profil zur Verfügung stellen, welches an die individuellen Aufgaben angepasst ist.

Durch diese Maßnahme wird der Benutzer von der totalen Profildefinition entlastet und muss „nur“ noch eine Feinabstimmung des Profils vornehmen. Von der Aktualisierung des

Profils befreit diese Methode den Anwender jedoch nicht.

3.2.4.2 Automatische Profilerzeugung

Die Ausführungen im vorigen Abschnitt machen deutlich, dass eine manuelle Profilerstellung nicht wünschenswert ist, sondern Mechanismen geschaffen werden müssen, die Profile automatisch erstellen oder zumindest bei der Profilgenerierung unterstützend mitwirken.

Damit Profile automatisiert erstellt werden können, muss sich ein solches System Informationen über die Person beschaffen, für die das Profil erstellt werden soll. Aus diesem Grund muss es Möglichkeiten geben, wie Daten über den Benutzer in das Modell gelangen können. Folgende Beispiele stellen Möglichkeiten hierfür dar:

- Texteingaben und Emails analysieren,
- Browsernavigation, URLs und Bookmarks beobachten,
- Resultate von Datenbankanfragen untersuchen,
- Besuchte Newsgroups und Chats analysieren sowie
- Intelligente Sensorik verwenden (Rückschlüsse auf die Umgebung einer Person oder den Gemütszustand der Person, Gesundheitszustand etc.).

Die Aufzählung zeigt, dass sämtliche Benutzerinteraktionen zur Profilgenerierung verwendet werden können. Aber auch zusätzliche Hardware (z. B. GPS-, Beschleunigungssensor) kann Informationen über den Benutzer oder dessen Eigenschaften liefern.

Die Problematik mit mobilen Geräten ist jedoch, dass die Anzahl der Benutzereingaben im Verhältnis zum Arbeitsplatzrechner reduziert ist. Eine Person schreibt mit einem PDA derzeit weder Briefe noch sucht sie intensiv im Internet nach Informationen. Auch die Email Korrespondenz ist gegenüber einem normalen Arbeitsplatzrechner geringer. Somit ist die Aufzeichnung von Benutzerinteraktionen zur Profilgenerierung für mobile Geräte nur bedingt gegeben.

Aus diesem Grund müsste die Profilerzeugung und Profilaktualisierung auf dem Arbeitsplatzrechner einer Person durchgeführt werden. Eine Übertragung der Profildaten auf das mobile Endgerät kann mit einer Synchronisation erfolgen.

Im Folgenden wird eine automatische Profilgenerierung mit Lernverfahren und eine Profilübernahme näher betrachtet.

Profilgenerierung durch Lernverfahren: In der Literatur werden zur automatischen Profilgenerierung verschiedene Arten von Lernalgorithmen verwendet.

Widyantoro *et al.* [WIY01] beschreiben ein Verfahren, wie Benutzerprofile (Präferenzen) gelernt werden können. Als Beispiel wird das Lernen von Dokumenten (z. B. aus dem Internet) herangezogen. Aus einem Dokument **D** wird ein so genannter *‚Feature-Vektor ‘* erzeugt. Grundlage hierfür sind charakteristische Wörter t_i, welche gehäuft in **D** auftauchen. Aus deren Häufigkeit in **D** und deren relative Häufigkeit in Standardtexten, wird für jedes Wort t_i ein Gewicht w_i errechnet [SM83]. Das Dokument **D** kann schließlich als $\{(t_1, w_1), (t_2, w_2), \ldots, (t_n, w_n)\}$ dargestellt werden.

Soltysiak und Crabtree [SC98] setzen sich zum Ziel mit minimalem *User Input* Benutzerprofile zu erstellen. Grundlage hiefür sind Dokumente, die der Benutzer selber schreibt oder liest. Darüber hinaus werden zur Profilerstellung Emails und die besuchten Webseiten analysiert. Diesen Dokumenten wird ein Vektor zugeordnet und es wird versucht ähnliche Vektoren zu finden und diese zu einem Cluster zusammenzufügen. Solch ein Cluster repräsentiert anschließend einen Eintrag im Benutzerprofil.

Bei der Klassifikation eines neuen potenziellen Eintrags wird die Ähnlichkeit mit Elementen aller bereits existierender Cluster überprüft. Bei „ausreichender“ Ähnlichkeit wird das neue Dokument sofort dem entsprechenden Cluster hinzugefügt. Ist jedoch für keinen Cluster die „genügende“ Ähnlichkeit gegeben, wird ein neuer Cluster erzeugt und die Elemente anderer Cluster werden überprüft, ob sie diesem neuen Cluster ähnlicher sind als ihrem momentan angehörigen. Ist nicht ersichtlich, ob dieser neue Cluster zu den Interessen oder Abneigungen des Benutzers gehört, wird erst am Ende des Klassifizierungsprozesses der Anwender mit einbezogen.

Auf Seiten des Benutzers liegt häufig eine Interessensverschiebung vor, d. h. die Interessen einer Person unterliegen Änderungen und Schwankungen. Ist eine Person im Sommer noch ein fanatischer Fußballanhänger, mutiert sie im Winter zum Skisprungenthusiasten. Folglich muss sich sein Profil diesen Interessenswechseln anpassen. Zu erwähnen ist, dass dieser Wechsel spontan oder schleichend vor sich gehen kann. Ein adaptives Klassifikationsverfahren sollte in der Lage sein sich diesen Änderungen anzupassen.

Diese Problematik behandeln Crabtree und Soltysiak [CS98]. Das Bilden von Dokument-Vektoren und Generieren von Clustern erfolgt gemäß dem früheren Verfahren [SC98]. Hinzu kommt jedoch eine Clusterhierarchie. Ausgehend von Dokumenten, die in Clustern zu Interessen gebündelt werden, werden diese wiederum zu *Themengebieten* zusammengefasst. Auf diese Weise wird eine Hierarchie Dokumente - Interessen - Themengebiete erzeugt. Um festzustellen, ob sich ein *Interessenwechsel* vollzogen hat, werden sich überlappende Zeitfenster für die Cluster verwendet. Die Fenster werden verglichen, wobei die Qualität der Übereinstimmung als indirekt proportional zur Stärke des Interessenswechsels angenommen wird.

Profilgenerierung durch Profilübernahme: Eine weitere Möglichkeit Profileinträge automatisch zu beziehen, ist die Übernahme von Profilteilmengen von anderen Personen. Hierzu ist jedoch schon ein gewisser „Kern“ an bereits vorhanden Profileinträgen nötig, die der Benutzer beispielsweise manuell definiert hat. Ist dieser Kern vorhanden, so kann das Profil mit dem Profil anderer Personen, die sich in momentaner Kommunikationsreichweite befinden, verglichen werden. Sind sich zwei Profilkerne (oder auch Profilteilmengen im späteren Fall) sehr ähnlich, so können Teile des Profils übernommen werden. Inwiefern die fremden Profilteile auf die Person zutreffen, muss getestet werden. Liegt ein hierarchisches Profil vor, bietet es sich an, primär Einträge von den konkreteren Ebenen zu übernehmen, da die Wahrscheinlichkeit, dass kleinere Neuerungen akzeptiert werden, größer ist als völlig neue Interessenszweige. Die neuen Einträge sind zuerst als tentativ markiert. Benutzer-Feedback entscheidet, ob die Einträge in das Profil eingetragen oder gelöscht werden.

3.2.5 Profile und Datenschutz

Generell sind Sicherheitsaspekte bei der Personalisierung von höchster Bedeutung. Niemand wäre erfreut, seine persönlichsten Daten in einem ihm unerwünschten Umfeld wieder zu sehen. Da im MoPiDiG Szenario Profile ausgetauscht werden, besteht permanent die Gefahr, dass Daten missbraucht werden.

Astor [Ast02] schlägt in MOTOMM die Einführung von Anonymitätstemplates vor, um die Profilinformation schrittweise zu verschleiern. In MOTOMM verfügt ein Benutzer über ein Profil, das sich aus seinen Stammdaten, seinen Präferenzen und seinen Anonymitätstemplates zusammensetzt. Die Stammdaten enthalten alle Angaben zur Person des Kunden. Sie sind die Basis für jedes persönliche Angebot und jede persönliche Nachfrage des Kunden. Die Präferenzen sind persönliche Einstellungen des Kunden. Die Anonymitätstemplates sind Schablonen auf die Stammdaten, d. h. sie gewähren eine Sicht auf einen Ausschnitt der Stammdaten. Jeder Kunde hat ein Satz von vordefinierten, d. h. nicht änderbaren Schablonen. Jeder Anwender kann zudem noch seine eigenen Schablonen definieren.

Auf diese Weise erreicht Astor, dass nicht permanent das komplette Profil für andere sichtbar ist, sondern nur Ausschnitte, die für die momentane Situation ausreichend sind. Der Teilbereich ist jedoch für alle Teilnehmer im Netzwerk sichtbar. Soll das Teilprofil nur für auserwählte Kommunikationsteilnehmer zugänglich sein, sind zusätzlich Authentifizierungsmaßnahmen nötig. Hierzu ist zu sagen, dass asymmetrische Verfahren, wie RSA [RSA78] nur bedingt geeignet sind. Zur Verschlüsselung von Nachrichten oder Profilen würde ein öffentlicher Schlüssel benötigt, der jedoch nicht bekannt sein muss, da Kommunikationspartner ebenfalls nicht bekannt sind.

3.3 Gruppenbildung

Das dieser Arbeit zu Grunde liegende Problem der Gruppenbildung wurde bereits in der Vergangenheit in der Literatur diskutiert.

Zuerst werden allgemeine Verfahren aus verschieden Gebieten der Informatik vorgestellt. Ein wesentlicher Bereich sind hier die so genannten Clustering-Verfahren. Unter Clustering werden Methoden verstanden, mit denen es möglich ist zusammengehörige Objekte aus einer Datenmenge zu finden.

Der nächste Abschnitt konzentriert sich auf bereits existierende Arbeiten des Problems der Gruppenbildung in ad hoc Netzwerken. Hierbei werden vor allem Unterschiede in der Aufgabenstellung und den Annahmen besonders hervorgehoben.

3.3.1 Gruppenbildung mit Clustering-Verfahren

Ein Cluster bzw. eine Gruppe ist im Folgenden eine Menge von Objekten, die untereinander eine hohe Ähnlichkeit aufweisen. Wie diese Cluster aussehen können ist beispielhaft in Abbildung 3.8 zu sehen. Dort sind in einem zweidimensionalen Profilraum verschiedene Formationen von Gruppen dargestellt. Abbildung 3.8a stellt die einfachste Methode dar. Hier werden n-dimensionale Kugeln als Gruppenformation benutzt. Für den zweidimensionalen Fall entspricht dies einem Kreis und für den dreidimensionalem Fall einer Kugel.

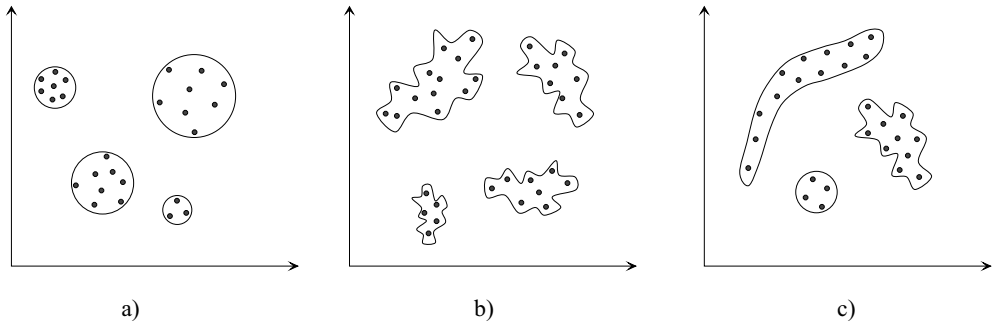


Abbildung 3.8: a) zeigt sphärische Cluster unterschiedlicher Größe. b) zeigt Cluster gleicher Form unterschiedlicher Orientierung. c) zeigt Cluster mit völlig willkürlicher Form und Größe.

Die Einfachheit resultiert aus der geschlossenen mathematischen Darstellbarkeit der Sphären und deren Symmetrie. Abbildung 3.8b stellt Gruppen mit nicht-sphärischen Formen und unterschiedlichen Größen dar. Die schwierigste Form stellt Abbildung 3.8c dar. In diesem Fall sollen die Gruppen willkürliche Form und Größe besitzen.

Bei allen dargestellten Formen werden ähnliche Punktwolken zu einer Gruppe zusammengefasst. Um diese Ähnlichkeit ausdrücken zu können, werden zuerst Verfahren vorgestellt, die dieses bewältigen können. Danach werden verschiedene allgemeine Methoden zur Gruppenbildung präsentiert.

3.3.1.1 Proximitätsmaße

Clustering fügt eine Menge von Objekten zu Gruppen zusammen, die einander „ähnlich“ sind. Deswegen ist ein Mechanismus notwendig, der die Ähnlichkeit von Objekten ausdrücken kann. Prinzipiell kann dies durch zwei Arten geschehen. Es kann angegeben werden, wie sehr sich zwei Objekte ähneln oder wie stark sie differieren. Erstere werden als Ähnlichkeitsmaße, die anderen als Distanzmaße bezeichnet. Der Überbegriff, der beide vereinigt, wird in der Literatur als Proximitätsmaß angegeben.

In Benutzerprofilen können die einzelnen Attribute unterschiedliche Wertebereiche aufweisen. Primär sind *nominalskalierte* und *intervallskalierte* Merkmale zu unterscheiden. Ein Merkmal bzw. die zugehörige Variable heißt *nominalskaliert*, wenn die Ausprägungen Namen oder Kategorien sind, die keine lineare Ordnung aufweisen, z. B. die Farbe eines Objekts ist rot, gelb oder grün. Den Ausprägungen werden dennoch (natürliche) Zahlen zugeordnet, die jedoch lediglich der Kodierung dienen und keine numerischen Werte im üblichen Sinne sind. Wenn die Ausprägungen linear geordnet werden können und die Differenzen zwischen den Ausprägungen eine einheitliche Interpretation besitzen, liegen *intervallskalierte* Merkmale vor, z. B. Skalenwerte.

Im Folgenden werden *Ähnlichkeits-* und *Distanzmaße* mathematisch definiert und jeweils Beispiele hierfür angegeben. Mit beiden Maßen ist die Bestimmung der Ähnlichkeit zweier Objekte möglich. Unterschied ist jeweils nur der Wertebereich für die Ähnlichkeits-

angabe. Bei den Ähnlichkeitsmaßen beschränkt sich dieser auf das Intervall $[0;1]$, während bei den Distanzmaßen das Intervall $[0;\infty]$ als Wertebereich gilt. Grund für die zwei analogen Ausdrücke ist, dass manche Algorithmen Distanzmaße und andere Ähnlichkeitsmaße verwenden, was zu einem späteren Zeitpunkt ersichtlich wird.

Ähnlichkeitsmaße: Durch Ähnlichkeitsmaße wird die Nähe zwischen Objekten im n-dimensionalen Merkmalsraum bestimmt. Ähnlichkeit im topologischen Sinne heißt, dass zwei Personen im Merkmalsraum der gemeinsamen Merkmale an einander nahe gelegenen Punkten platziert sind.

Definition 3: *Ähnlichkeitsmaß* $sim(x, y)$
Sei M ein Merkmalsraum, dann ist ein Ähnlichkeitsmaß eine Abbildung sim: $M \times M \rightarrow [0;1]$, falls für alle n-dimensionalen Vektoren x, y $\in$ M gilt:

$$sim(x,x) = 1 \qquad \text{(Reflexivität)} \tag{3.7}$$
$$sim(x,y) = sim(y,x) \qquad \text{(Symmetrie)} \tag{3.8}$$
$$sim(x,y) = 1 \Leftrightarrow x = y \tag{3.9}$$

□

Ein Beispiel für ein Ähnlichkeitsmaß ist der Cosinus-Abstand:

$$sim(x,y) = \cos\alpha(x,y) = \frac{x \cdot y}{|x||y|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

Hier seien x, y Vektoren und $|x|$, $|y|$ deren Beträge, dann entspricht der Winkel, den diese beiden Vektoren zueinander bilden, dem Maß für die Ähnlichkeit. Um den Anforderungen an die Definition des Ähnlichkeitsmaßes gerecht zu werden, wird der Winkel durch den Kosinus des Winkels ausgedrückt.

Distanzmaße: Durch Distanzmaße wird die Entfernung zwischen Objekten im n-dimensionalen Merkmalsraum bestimmt. Voraussetzung für die Verwendung von Distanzmaßen ist das Vorliegen intervallskalierter Daten, bzw. eine entsprechende Modellierung der Daten.

Definition 4: *Distanzmaß*
Eine Abbildung d: $M \times M \rightarrow \mathbb{R}^+$ heißt Distanzmaß, falls für alle n-dimensionalen Vektoren x, y $\in$ M gilt:

$$\begin{aligned} d(x,y) &= 0 \Leftrightarrow x = y && \text{(Definitheit)} \\ d(x,y) &= d(y,x) && \text{(Symmetrie)} \\ d(x,z) &\leq d(x,y) + d(y,z) && \text{(Dreiecksungleichung)} \end{aligned}$$

□

Aus der Fülle der existierenden Distanzfunktionen soll hier die Minkowski Distanz $d_n(x,y)$, mit

$$d_n(x,y) = \sqrt[n]{\sum_{i=1}^{m} |x_i - y_i|^n} \qquad (n \in \mathbb{N}) \tag{3.10}$$

Erwähnung finden. Der Grund hierfür ist, dass folgende Spezialfälle der Minkowski Distanz sehr häufig Verwendung finden:

$$d_1(x,y) = \sum_{i=1}^{m} |x_i - y_i| \tag{3.11}$$

$$d_2(x,y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \tag{3.12}$$

$$d_\infty(x,y) = \max_i |x_i - y_i| \tag{3.13}$$

$d_1(x,y)$ wird als Manhattan-Distanz[5] bezeichnet. Die Manhattan Distanz ist robuster gegenüber Ausreißern als die Euklidische Distanz $d_2(x,y)$, welche den geometrischen Abstand zweier Punkte x und y im Raum widerspiegelt. $d_\infty(x,y)$ wird als Maximum-Distanz bezeichnet und reduziert den Abstand zweier Punkte auf deren größte Abweichung in einer Dimension.

Für den Fall, dass die zu gruppierenden Objekte nicht aus $\mathbb{R}^n$ stammen, sondern nominale Attribute vorliegen, wird folgende Distanzfunktion verwendet:

$$d_A(x,y) = \sum_{i=1}^{m} \delta(x_i, y_i) \qquad \text{mit } \delta(x_i, y_i) = \begin{cases} 0 & \text{falls } x_i = y_i, \\ 1 & \text{sonst.} \end{cases} \tag{3.14}$$

Hier wird die Anzahl an verschiedenartigen Einträgen gezählt: Je mehr Unterschiede es gibt, desto größer ist die Distanz.

Für den Anteil verschiedener Elemente in Mengen in x und y bietet sich folgende Distanzfunktion an:

$$d(x,y) = \frac{|x \cup y| - |x \cap y|}{|x \cup y|} \tag{3.15}$$

Diese Distanzfunktion beschreibt das Verhältnis von Objekten, die in nur einer Menge vorkommen, zur Gesamtanzahl der Objekte in beiden Mengen.

Eine sehr einfache Art Distanzen zwischen Elementen auszudrücken, stellen Distanz-Matrizen $\mathbf{D}_{ij}$ (Proximity Matrix) dar.

$$\mathbf{D} = \begin{pmatrix} d_{11} & d_{12} & \dots & 1m \\ d_{21} & \ddots & & \\ \vdots & & \ddots & \\ d_{m1} & & & d_{mm} \end{pmatrix} \tag{3.16}$$

In der Matrix 3.16 kann zwischen je zwei Elementen die Distanz direkt angegeben werden. Der Vorteil dieses Verfahrens ist, dass keine Funktion existieren muss, die für alle angegebenen Distanzen gültig ist.

Abschließend sei noch bemerkt, dass es nicht *die* Distanzfunktion gibt, die für jede Anwendung einsetzbar ist. Vielmehr muss für jede Anwendung nach der passendsten Distanzfunktion gesucht werden, denn deren Wahl ist ausschlaggebend für das Ergebnis.

[5] Die Bezeichnung *Manhattan-Distanz* ist im zwei-dim. Raum dem Bild rechteckig angelegter Straßenzüge angelehnt. Die Entfernung zweier Punkte ist hier nicht durch die Luftlinie gegeben, sondern durch Entlangfahren an Straßenzügen.

3.3.1.2 Partitionierendes Clustering

Partitionierendes Clustering [Fas99] hat als Initialparameter eine Menge S an Objekten und einen positiven, ganzzahligen Wert k, der die Anzahl an Partitionen angibt. Ergebnis ist anschließend eine Partitionierung der Menge S in Teilmengen $S_1, S_2, \ldots, S_k$, mit $S = \bigcup_{n=1}^{k} S_k$.

k-Means-Algorithmus: Der k-means Algorithmus [Fas99] ist einer der verbreitetsten Clusteralgorithmen, weil ihm ein einfaches Prinzip zugrunde liegt und er gute Ergebnisse liefert. Der k-Means-Algorithmus behandelt das Clusterproblem algorithmisch als ein Optimierungsproblem. Mit jedem Cluster S_i wird der Wert einer Kostenfunktion $c(S_i)$ verbunden. So sucht man nach einem Minimum von $\sum_{i=1}^{k} c(S_i)$. Eine oft verwendete Kostenfunktion ist

$$c(S_i) = \sum_{r=1}^{|S_i|} \sum_{s=1}^{|S_i|} (d(x_r^i, x_s^i))^2 \tag{3.17}$$

Hierbei ist x_j^i das j-te Element von S_i und $d(x_j^i, x_k^i)$ die Distanz (siehe mögliche Funktionen in Abschnitt 3.3.1.1) zwischen x_j^i und x_k^i. Neben Gleichung 3.17 sind auch andere Kostenfunktionen möglich.

Doch sind mit dem Algorithmus auch Probleme verbunden. So ist die komplette Optimierung NP-hart. Deswegen werden bei seiner Anwendung oft nur lokale Optima gesucht. Hierzu existieren Lösungen, die schrittweise das Endergebnis verbessern. Doch auch hierbei ist Vorsicht geboten, da diese nicht mehr eindeutig sein müssen. Außerdem sind die Cluster, die mit ihm bestimmt werden können konvex, d. h. beliebige Formen wie in Abbildung 3.8c sind nicht möglich. Da der Algorithmus sämtliche Eingabedaten in die Partitionierung mit einbezieht, ist es ratsam Ausreißer vorher zu entfernen.

Fuzzy-Clustering: Durch kleine Änderungen des vorhergehenden Algorithmus wird ein *Fuzzy-Clustering* erreicht [MD96]. Im Gegensatz zu scharfen Cluster-Verfahren, bei denen jedes Datum genau einem Cluster zugeordnet wird, erhalten die Daten beim fuzzy (unscharfen) Clustering für jedes Cluster einen Zugehörigkeitsgrad aus dem Intervall $[0, 1]$, der angibt, wie gut das betreffende Datum zum jeweiligen Cluster passt. Im Allgemeinen werden andere Kostenfunktionen als Gleichung 3.17 verwendet. Auch diese Variante ist empfindlich gegenüber Ausreißern, so dass deren vorherige Entfernung anzuraten ist.

3.3.1.3 Hierarchisches Clustering

Bei dieser Art des Clusterns ist ebenfalls eine Menge S von Objekten als Eingabeparameter vorgesehen. Beim hierarchischen Clustern [Fas99] wird aus der Menge S ein Baum (auch Dendogramm genannt) $T(S)$ erzeugt, dessen Knoten die einzelnen Teilmengen von S und dessen Blätter die Elemente von S repräsentieren. Der Vorteil dieses Verfahrens ist Cluster verschiedener *Genauigkeit* zu erhalten. Sind nur die groben Eigenschaften von Gruppenmitgliedern von Interesse, sind nur die oberen Ebenen des Baumes in Betracht zu ziehen. Je tiefer von der Wurzel zu den Blättern traversiert wird, desto spezifischer werden schließlich die Cluster. Die Anzahl der Cluster wird - im Gegensatz zum Partitionierenden Clustering - erst zur Laufzeit bestimmt. Es gibt prinzipiell zwei Methoden des

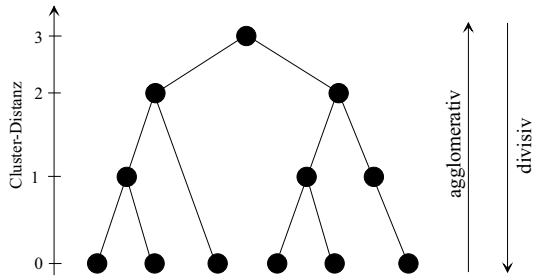


Abbildung 3.9: Dendogramm für hierarchisches Clustering

hierarchischen Clusterns - *divisive* und *agglomerative* Algorithmen. Divisive Algorithmen gehen von einem einzigen Cluster aus, in dem alle Elemente enthalten sind (i. A. die Eingabemenge S) und entfernen sukzessive Elemente. Es wird jeweils das Element entfernt, welches den größten Abstand zu den restlichen Elementen aufweist. Dieser Vorgang wird so oft wiederholt, bis jeweils nur Cluster mit einem Element vorhanden sind.

Agglomerative Algorithmen gehen den konträren Weg und beginnen mit so vielen Clustern, wie es Elemente gibt und versuchen sukzessive die jeweils ähnlichsten Clusterpaare zu einem zu verbinden. Eine Kostenfunktion gibt an, ob zwei Mengen S_i und S_j zusammengeführt werden sollen oder nicht. Abbildung 3.9 zeigt anhand eines Dendogrammes noch einmal das Prinzip der Algorithmen. Auf der untersten Ebene, den Blättern des Baumes, ist die Distanz gleich null, da ein Cluster nur aus einem Element besteht. Nach oben hin nimmt sowohl die Distanz als auch die Gruppengröße zu.

3.3.1.4 Wahrscheinlichkeitsbasiertes Clustering

Als wahrscheinlichkeitsbasierte Verfahren sind primär die dichtebasierten Methoden wie z. B. die *Mixture Models* zu nennen [Fas99]. Die Eingabevektoren $\mathbf{x}_i$ stammen bei diesem Verfahren aus einer Menge von k unbekannten Wahrscheinlichkeitsverteilungen E_1,-$E_2, \ldots, E_k$. Die Dichtefunktion der Beobachtung $\mathbf{x}_r$ unter der Verteilung E_i sei $f_i(\mathbf{x}_r|\theta)$. θ sei die Menge der unbekannten Parameter. Ferner sei für jedes $\mathbf{x}_r$ und die Verteilung E_i, τ_r^i die Wahrscheinlichkeit, dass $\mathbf{x}_r$ zu E_i gehört. Um ein Ergebnis zu erzielen, gilt es

$$L(\theta, \tau) = \prod_{r=1}^{n} \sum_{i=1}^{k} \tau_r^i f_i(\mathbf{x}_r|\theta)$$

zu maximieren, indem θ und τ variiert werden.

Banfield and Raftery [BR93] konzentrieren sich bei den Verteilungen auf die Normalverteilung. In diesem Fall sind die unbekannten Parameter θ, der Mittelwert-Vektor μ_i und die Kovarianz-Matrix $\mathbf{\Sigma}_i$. Die Autoren variieren in ihrem Ansatz die Kovarianzmatrix, wobei sie von Diagonal-Matrizen ausgehen. Auf diese Weise transformieren sie die Lösung in ein Problem der linearen Algebra.

Bezüglich der Anwendbarkeit dieser *Mixture Models* sagt Maitra [Mai98], dass diese sich für große Datenmengen besser eignen als beispielsweise der k-means Algorithmus aus Abschnitt 3.3.1.2.

3.3.1.5 Clustering für hochdimensionale Daten

Das Clustering Problem wird noch komplexer, wenn die Anzahl der Dimensionen der Merkmalsvektoren ansteigt. In diesem Fall kommt es oft vor, dass die Daten extrem im Raum verteilt sind und auf allen Dimensionen kein Cluster gefunden werden kann. Ein Beispiel hierfür ist in Abbildung 3.10 zu sehen.

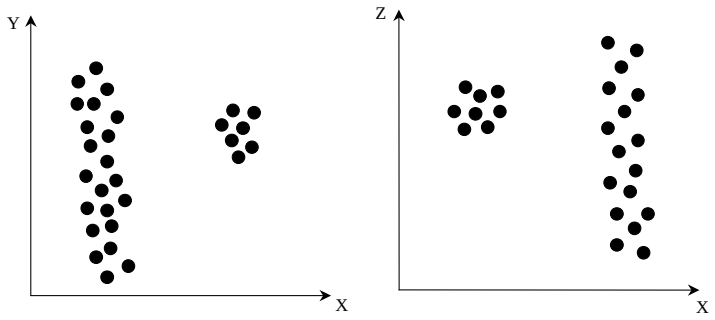


Abbildung 3.10: Probleme bei der Clusterfindung bei mehreren Dimensionen (in Anlehnung an Aggarwal *et al.* [AWY+99])

Es müssen nicht alle Dimensionen für ein Clustering betrachtet werden. Die Aufgabe besteht darin, geeignete Subräume zu finden, in denen Cluster existieren. Was jedoch die Anzahl dieser Subräume anbelangt, so ist sie tendenziell sehr groß. Für eine gegebene Dimension d, gibt es insgesamt 2^d Teilräume (Potenzmenge der Dimensionen), die zu untersuchen wären. Eine reine Suche in den Räumen scheidet aus diesen Gründen aus. Auch eine Vorauswahl an Dimensionen für ein Clustering ist nicht ausreichend, da in verschieden Fällen unterschiedliche Dimensionen für ein Clustering herangezogen werden müssen.

Aggarwal *et al.* [AWY+99] beschreiben eine Clustering-Variante namens „*Projected Clustering*“ . Ein *Projected Cluster* ist eine Teilmenge $\mathcal{C}$ der Daten und eine Teilmenge $\mathcal{D}$ der Dimensionen derart, dass die Punkte in $\mathcal{C}$ in $\mathcal{D}$ Cluster bilden. Abbildung 3.10 zeigt jeweils zwei projizierte Cluster in der x-y-Ebene und x-z-Ebene.

Der beschriebene Algorithmus PROCLUS (PROjected CLUStering) benötigt als Eingabeparameter die Anzahl k der zu findenden Cluster und die durchschnittliche Dimension l eines solchen Clusters. Ergebnis sind $(k + 1)$ Partitionen und die dazugehörigen Dimensionen. Die zusätzliche Partition (es sollten eigentlich nur k sein) ist kein Cluster, sondern nur die Menge aller Ausreißer. Als Distanzfunktion wird die Manhattan-Distanz verwendet. Soll nach Cluster in höheren Dimensionen gesucht werden, müssen Clusterzentren und Dimensionen gefunden werden, in denen diese existieren.

Aggarwal [Agg01] beschreibt noch eine andere Möglichkeit, Cluster in hoch dimensionalen Räumen zu finden, nämlich mit Hilfe von menschlicher Interaktion. Hier wird das komplexe Clusterproblem mit menschlicher Intuition zu lösen versucht. Die Idee hinter IPCLUS (Interactive Projected CLUStering) ist es, dem Benutzer, der mitentscheiden soll,

wie die Cluster weiter gebildet werden sollen, disjunkte niedrig dimensionale Projektionen vorzulegen.

3.3.1.6 Genetische Algorithmen

Auch mit Hilfe von *Genetischen Algorithmen* können Gruppen erzeugt werden, wie es beispielsweise Tseng und Yang [TY00] erläutern. Es wird ein genetischer Algorithmus beschrieben, der in der Lage ist nicht-sphärische Cluster zu generieren. Außerdem wird die Anzahl an Clustern nicht wie beim k-Means Algorithmus als Parameter benötigt.

Das Verfahren bildet Cluster auf (0/1)-Zeichenketten ab. In den einzelnen Evolutionen des Algorithmus werden durch einen zufallsbasierten Ansatz die Strings geändert und überprüft, ob sich die Clustereigenschaften verbessert haben. Der Algorithmus hat laut Tseng und Yang eine Laufzeit von $\mathcal{O}(n^2)$.

Der Algorithmus leidet unter einer schlechten Nachvollziehbarkeit, d. h. es ist nur schwer ersichtlich, weshalb das Verfahren konvergiert und die entstandenen Cluster optimal sein sollen.

3.3.2 Gruppenbildung in ad hoc Netzwerken

Im vorherigen Abschnitt wurden Verfahren vorgestellt, deren Ursprung im Datanbankbereich liegt und deren Anwendbarkeit im Rahmen von MoPiDiG auf das mobilen Umfeld überprüft wird. In diesem Abschnitt werden existierende Arbeiten zur Gruppenbildung in ad hoc Netzwerken präsentiert.

Es gibt verschiedene Möglichkeiten, wie und welche mobilen Endgeräte sich zu Gruppen zusammenschließen. So zeigt Abbildung 3.11a eine gewöhnliche N-Hop-Gruppe.

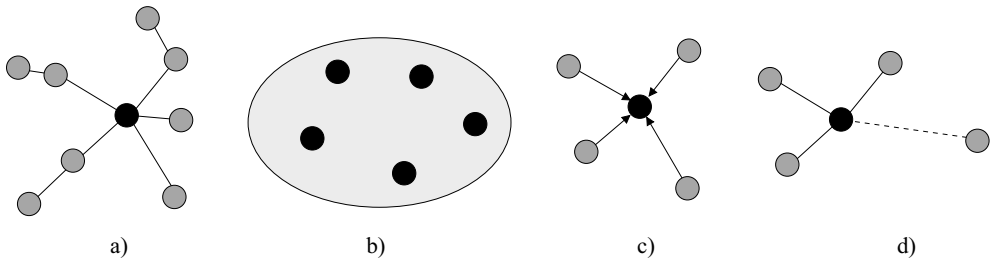


Abbildung 3.11: Verschiedene Möglichkeiten der Gruppenbildung.

Hier schließen sich alle Geräte zu einer Gruppe zusammen, die höchstens N Hops voneinander entfernt sind. Die zweite Möglichkeit ist es, eine Gruppe an einen festen Ort zu platzieren. So sind in Abbildung 3.11b alle Elemente innerhalb der Ellipse Gruppenmitglieder. Eine Realisierung könnte beispielsweise mit Basisstationen erfolgen, wie sie in Abschnitt 3.1.1 eingeführt wurden. Abbildung 3.11c stellt eine Publish/Subscribe-Gruppe [WQA+02] dar. In dieser veröffentlicht ein Gerät (der Publisher) spezielle Informationen, die die Grundlage für die Gruppenbildung darstellen. Andere Benutzer (die Subscriber) schließen sich dem Publisher an. Die letzte Variante (Abbildung 3.11d) ist eine Erweiterung der N-Hop-Gruppe, wobei hier zusätzlich die Verbindung erhalten bleibt, wenn sich

ein Gerät aus dem N-Hop Bereich entfernt. Für die Realisierung muss eine zusätzliche, weit reichende Kommunikationsmöglichkeit (z. B. GSM) existieren, um die Verbindung unabhängig von anderen Geräten zu erhalten.

Roman *et al.* [RHH01] beschäftigen sich mit mobiler Gruppenfindung in ad hoc Netzwerken, damit die Mitglieder beispielsweise gemeinsame Anwendungen benutzen können. Um die Probleme, die durch die Mobilität auftreten zu beherrschen, führen die Autoren eine *Safe Distance* ein. Diese definiert einen oberen Schwellwert für die Entfernung zweier Kommunikationspartner, sodass eine Kommunikation gestartet werden kann und keine Gefahr besteht, dass die Kommunikation unterbrochen wird, weil sich ein Kommunikationsteilnehmer aus dem Kommunikationsradius bewegt. Die *Safe Distance* hängt von der Geschwindigkeit der Gerätebesitzer und vom Kommunikationsradius ab. Dieses Konzept schützt zwar vor Verbindungsunterbrechungen, die auf die Mobilität zurückzuführen sind, durch Ausfall oder Abschalten des Gerätes können diese Probleme - wenn auch nicht mehr so häufig - jedoch trotzdem noch auftreten.

Ferner existiert für jede Gruppe ein Gruppenführer, der die Aufenthaltsorte seiner Gruppenmitglieder kennt und aktualisiert. Dieser Führer entscheidet, ob zwischen zwei potenziellen Kommunikationspartnern eine *Safe Distance* vorliegt oder nicht.

Architektonisch unterteilen Roman *et al.* ihr Verfahren in ein *Group Discovery Protocol*, welches benachbarte Geräte zu Gruppen zusammenführt und ein *Reconfiguration Protocol*, das für Gruppenaufteilung und Gruppenverschmelzung zuständig ist.

Abschießend soll bewertend erwähnt werden, dass sich die in Roman *et al.* beschriebenen Konzepte anbieten, wenn Ortsinformation verfügbar ist. Die Einführung eines Gruppenführers ist in mobilen Anwendungen jedoch generell als kritisch zu betrachten, da dieser ausfallen kann und anschließend erst mühsam ein neuer Führer erwählt werden muss. In dieser Zeit ist die Gruppe i. A. handlungsunfähig.

Meissner und Musunoori [MM03] gehen näher auf Gruppenmanagement in ad hoc Netzwerken ein. Für die Autoren ist eine Gruppe eine Ansammlung von Leuten, die zu einem Zeitpunkt miteinander kommunizieren können. Als Anwendungsfälle werden Informationsaustauschszenarien wie der Tausch von Abflugterminen und Sitzplätzen an Flughäfen präsentiert. Aber auch der sinnvolle Einsatz von Gruppen bei der Hilfe nach Naturkatastrophen wird genannt.

In dem Framework werden die Gruppen von einem *Manager* verwaltet. Dieser entscheidet auf Basis bestimmter Bedingungen, ob ein Benutzer in die Gruppe mit aufgenommen wird. Die Autoren schlagen für diesen Manager sowohl eine separate zentrale Komponente vor als auch die Übernahme dieser Tätigkeit durch ein Gruppenmitglied. Im Gegensatz zu Roman *et al.* [RHH01] stellen die Autoren in ihrem Framework Ersatzmanager zur Verfügung für den Fall, dass der Erste ausfällt.

Koch *et al.* [KGH02] erweitern, wie auch später Groh [Gro03], den Terminus 'Gruppe' durch den Begriff *Community*. Eine *Community* ist eine Gruppe von Menschen, die gemeinsame Interessen haben und zusätzlich noch eine Kommunikationsmöglichkeit besitzen, um sich über diese Interessen auszutauschen. Waren bis zu diesem Zeitpunkt Communities nur im Festnetz (z. B. Internet) verbreitet, übertragen Koch *et al.* das bisherige Anwendungsgebiet der Communities auf mobile Endgeräte.

In der Lösung wird jedoch kein ad hoc Netzwerk verwendet. Die Kommunikation er-

folgt über das Mobilefunknetz (primär SMS). Hierdurch kann die nötige Ortsinformation - sofern benötigt - aus der Zelleninformation des Mobilfunknetzes entnommen werden.

Groh [Gro03] zeigt ferner Lösungsmöglichkeiten auf, wie Gruppen mit ähnlichen Interessen gefunden werden können. Der Autor schlägt hierfür die Methoden vor, die in Abschnitt 3.3.1.2 vorgestellt wurden. Alle Endgeräte senden ihre Daten an einen zentralen Server, der die Gruppenbildung vornimmt.

Astor [Ast02] beschreibt Mobiles One-To-One Matchmaking[6] (kurz: MOTOMM). Die Aufgabe von MOTOMM ist das personalisierte Vergleichen und Bewerten von Angeboten und Nachfragen im lokalen Umfeld mobiler Endgeräte. MOTOMM weist den Nutzer auf interessante, für ihn spezifisch zugeschnittene Angebote (respektive Nachfragen) in seiner unmittelbaren Umgebung hin und ermöglicht somit dem Benutzer zu dem vermittelten Anbieter Kontakt aufzunehmen.

Behandelt Astor eine 1:1 Kommunikation, kann MoPiDiG als Erweiterung zu diesem Gedanken gesehen werden, da in MoPiDiG eine 1:n Kommunikation vorliegt.

Badache *et al.* [BHM97] untersuchten als Erste das consensus Problem in mobilen Umgebungen. Sie beschränken sich jedoch auf ad hoc Netzwerke mit Infrastruktur. Die Basisstation dient als Datensammler, wobei auch berücksichtigt wird, dass diese ausfallen kann. Die Kommunikation wird von einem Koordinator gesteuert. Die Koordinatorrolle rotiert jedoch zwischen allen Teilnehmern. Da der Einigungsalgorithmus die Basisstationen mit einbezieht, ist er nicht auf infrastrukturlose ad hoc Netzwerke übertragbar.

Basagni [Bas99] beschreibt mit dem *Distributed and Mobility-Adaptive Clustering*-Algorithmus die Zerlegung eines ad hoc Netzwerkes in disjunkte Cluster. Der Autor beschäftigt sich intensiv mit der Mobilität der Knoten während der Gruppenerzeugung. Basagni benutzt den oben beschriebenen Clusterhead-Ansatz zum Clustering. Zu dessen Identifizierung wird der Ansatz des Lowest-ID Clustering verwendet. Basagni postuliert, dass bezüglich der Mobilität der Endgeräte *keine* Annahmen gemacht werden dürfen, d. h. es kann nicht davon ausgegangen werden, dass das Netzwerk während der Clustererstellung statisch ist. Somit können während des Prozesses neue Geräte ins ad hoc Netzwerk aufgenommen werden und andere verschwinden.

Der Algorithmus erlaubt ein Bewegen der Geräte während der Clustererzeugung und sorgt für den Erhalt der Gruppe. Einzige Annahme ist, dass eine gesendete Nachricht innerhalb zeitlicher Grenzen bei allen Kommunikationspartnern ankommen muss.

Heinemann [HKLMh03] beschreibt das Framework *iClouds*, welches eine Architektur für „spontane Kollaboration“ darstellt. Die Kollaboration findet in Gruppen mit gemeinsamen Interessen statt. Die *iClouds* Architektur ist auf keine zentralen Einheiten angewiesen. *iClouds* konzentriert sich auf Gruppen in der unmittelbaren Umgebung und schließt deswegen Multihop-Kommunikation aus. Kommunikation findet im infrastrukturlosen ad hoc Modus mit WLAN statt.

Die Daten, die die Grundlage für die Gruppenbildung darstellen, werden in zwei Listen verwaltet. Eine *iHave*-Liste stellt die Informationen dar, die der Benutzer seinen Gruppenmitgliedern zur Verfügung stellt, wobei die *iWish*-Liste definiert, an welchen Informationen

[6] Matchmaking ist das Vergleichen von Angebots- und Nachfrageprofilen und die Bewertung ihrer Ähnlichkeit bzw. Übereinstimmung [VMSF01].

er noch interessiert ist. Als Datenrepräsentation dient XML und zur Übereinstimmung der Listen wird auf eine Art Matchmaking zurückgegriffen. Somit ist der Ansatz ähnlich zu MOTOMM [Ast02].

In Tabelle 3.3 sind die existierenden Lösungen zur Gruppenbildung nochmals wieder-

	ad hoc Kommunikation	dezentraler Ansatz	Ortsinformation unnötig	kein Gruppenführer notwendig	Geschwindigkeit beschränkt	maximale Gruppengröße
Roman *et al.* [RHH01]	★	★			(★)	k. A.
Meissner und Musunoori [MM03]	★	(★)	★	(★)		k. A. k. A.
Koch *et al.* [KGH02]				★		k. A.
Groh [Gro03]			★	★		k. A.
Astor [Ast02]	(★)	(★)	★	★	k. A.	2
Badache *et al.* [BHM97]	(★)		★			k. A.
Basagni [Bas99]	★	★	★			k. A.
Heinemann [HKLMh03]	★	★	★	★	k. A.	k. A.
MoPiDiG [SBB04b]	★	★	★	★	★	anwendungs-abhängig

Tabelle 3.3: Vergleich existierender Lösungen für die Gruppenbildung in mobilen Umgebungen

gegeben, um sie direkt miteinander vergleichen zu können. Benötigt ein Framework eine gewisse Eigenschaft, so ist dies mit einem Stern gekennzeichnet. Eingeklammerte Sterne deuten auf hybride Ansätze hin. Hinsichtlich der Skalierbarkeit der Gruppengröße sind bei den jeweiligen Frameworks i. A. keine Angaben (k. A.) vorhanden.

Gemäß Tabelle 3.3 ist das MoPiDiG Framework gemäß den Eigenschaften dem iClouds-Framework von Heinemann [HKLMh03] sehr ähnlich. Jedoch differieren die Ziele der beiden Frameworks. Während iClouds explizit Listen vergleicht und auf diese Weise bei gleichen Einträgen Gruppen bildet, sucht MoPiDiG nach ähnlichen Daten ohne genaue Vorgabe bzw. stellt Gruppen nach gewissen Kriterien zusammen.

3.4 Zusammenfassung

In diesem Kapitel wird die Einordnung der Problembereiche erläutert, die mit dem MoPiDiG Framework verbunden sind.

Zuerst werden ad hoc Netzwerke hinsichtlich der vorhandenen Infrastruktur klassifiziert und erläutert. Durch die Mobilität der Teilnehmer eines ad hoc Netzwerkes kann die Kommunikation unterbrochen werden. Da dies mit der Größe des Netzes zunimmt, werden Methoden gezeigt, wie ein Netzwerk in mehrere Teilnetze segmentiert werden kann. Zur

Kommunikation in ad hoc Netzwerken stehen zur Zeit die Funktechnologien Bluetooth und WLAN zur Verfügung.

Für eine MoPiDiG Anwendung ist ein Benutzerprofil notwendig. Hinsichtlich der Profilerzeugung existieren manuelle aber auch automatische Verfahren, wobei im MoPiDiG Framework von einem bereits existierenden Mechanismus zur Profilerzeugung ausgegangen wird.

Für die Gruppenbildung können Verfahren aus der Clusteranalyse verwendet werden, mit welchen es möglich ist, aus einer Datensammlung „ähnliche“ Daten zu finden. Die Ähnlichkeit wird mit den der Anwendung angepassten Proximitätsmaßen formuliert.

Arbeiten zur Gruppenbildung in mobilen Umgebungen existieren ebenfalls, wobei sich die Anforderungen im Vergleich zum MoPiDiG Framework unterscheiden.

All die in diesem Kapitel genannten Verfahren können zur Lösung des MoPiDiG Frameworks heran- oder einbezogen werden. Hierbei ist jedoch eine Adaption der einzelnen Verfahren notwendig. Welche Methoden zur Lösung des MoPiDiG Szenarios verwendet werden, wie sie adaptiert werden müssen und wie die einzelnen Komponenten im Rahmen einer kompletten Architektur zusammen harmonieren, ist Gegenstand des nächsten Kapitels.

Kapitel 4

Das MoPiDiG Framework

In diesem Kapitel wird der generelle Aufbau des MoPiDiG Frameworks beschrieben, ohne den Fokus auf ein spezielles Anwendungsszenario zu richten.

Zuerst wird die Architektur von MoPiDiG präsentiert. Davon ausgehend wird in den folgenden Abschnitten jede Komponente der Architektur und deren Funktionsweise detailliert erläutert. Der Schwerpunkt der Darstellung liegt in der Beschreibung der Algorithmen, die zur Gruppierung notwendig sind. Das Kapitel endet mit einer kurzen Zusammenfassung.

4.1 Architektur des MoPiDiG Frameworks

In diesem Abschnitt werden der Aufbau und die einzelnen Komponenten des MoPiDiG Frameworks vorgestellt. Ferner werden die Annahmen angegeben, auf welchen das MoPiDiG Framework beruht.

4.1.1 Aufbau des MoPiDiG Frameworks

In diesem Abschnitt wird die Architektur des MoPiDiG Framework vorgestellt, welche in Abbildung 4.1 zu sehen ist.

Grundlage des MoPiDiG Frameworks ist eine Middleware. Diese stellt grundlegende Dienste wie die Benutzererkennung und Kommunikation zur Verfügung. Erwähnt werden muss, dass die Middleware nicht Entwicklungsbestandteil des MoPiDiG Frameworks ist, sondern hierbei auf bereits existierende Middleware zurückgegriffen wird.

Auf der Middleware ist der Kern des Frameworks, die MoPiDiG-Ebene, angesiedelt. Diese Schicht ist bei jeder MoPiDiG Anwendung identisch vorhanden, braucht keinerlei Anpassungen und ist somit domänenunabhängig. Diese Ebene besteht aus Algorithmen, welche den eigentlichen Gruppierungsprozess realisieren. Um das MoPiDiG Framework an keine Anwendung zu fixieren, existieren Meta-Beschreibungen, die die Domänenunabhängigkeit von MoPiDiG ermöglichen. Eine Meta-Beschreibung für Profileinträge spezifiziert, wie die Profilbeschreibung aufgebaut sein muss, um von der Algorithmuskomponente verwendet werden zu können. Da der Gruppenbegriff auch von Anwendung zu Anwendung variiert, werden in der Meta-Gruppenbeschreibung abstrakt die Charakteristika einer Gruppe festgelegt. Die Konsistenzüberprüfung stellt fest, ob ein gegebenes Profil

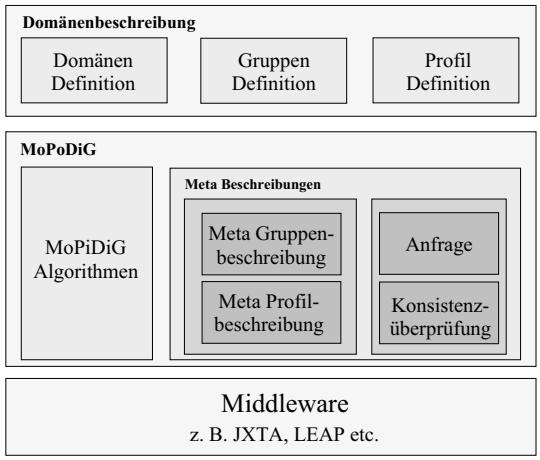


Abbildung 4.1: Aufbau des MoPiDiG Frameworks

den Meta-Beschreibungen entspricht und stellt ferner noch eine Anfrageschnittstelle für die Algorithmen an die Profile dar.

Die oberste Schicht des MoPiDiG Frameworks bildet die Domänenbeschreibung. Das Benutzerprofil einer Person ist in der Profildefinition abgelegt. Außerdem wird in der Gruppendefinition festgelegt, wie für ein bestimmtes Anwendungsgebiet eine Gruppe definiert ist. Die Domänendefinition ermöglicht die Konfiguration des Frameworks und bietet eine Programmierschnittstelle.

Zu beachten ist hierbei, dass die Meta-Beschreibungen aus der darunter liegenden Schicht nicht verletzt werden, was von der Konsistenzüberprüfung untersucht wird.

4.1.2 Annahmen und Anforderungen

Um das MoPiDiG Framework umsetzen zu können sind gewisse Annahmen und Anforderungen nötig, die in dieser Abschnitt kurz erläutert werden.

Identifikatoren

Bezüglich der Identifikation der an einer Gruppenbildung beteiligten Geräte wird gefordert, dass jedes Gerät über eine global eindeutige ID verfügt. Über die Repräsentation der IDs wird an dieser Stelle keine Anforderungen gestellt. Zu beachten ist jedoch, dass auf den IDs eine Ordnungsrelation (z. B. „$\leqslant$“) definiert werden kann, da diese Eigenschaft Bestandteil eines Algorithmus ist.

Im Weiteren wird gefordert, dass ein Gerät sämtliche IDs seiner direkten Kommunikationspartner kennt. Diese Anforderung wird i. A. jedoch von existierender Middleware erfüllt, die für ad hoc Netzwerke eingesetzt werden kann.

Annahmen bezüglich der Mobilität

Die Teilnehmer einer Anwendung des MoPiDiG Frameworks bewegen sich mit einer bestimmten Geschwindigkeit, die nicht bekannt sein muss. Bezüglich der Höhe der absoluten Geschwindigkeit werden keine Einschränkungen auferlegt, sofern ein Nachrichtenaustausch noch möglich ist. Jedoch wird gefordert, dass hinsichtlich der Relativgeschwindigkeit der Teilnehmer eine obere Grenze existiert, weil ab dieser oberen Geschwindigkeitsgrenze die Kommunikationszeit der Gruppenteilnehmer zu gering ist, um stabile Gruppen zu erzeugen. Die Obergrenze kann an dieser Stelle nicht pauschal angegeben werden, da sie auch vom Bewegungsprofil der Benutzer in einer gegebenen Domäne abhängt.

4.2 Middleware - Basis für das MoPiDiG Framework

Die Basis für das MoPiDiG Framework ist die Middleware. Unter Middleware sei hier eine Verbindung zwischen Betriebsystem und Anwendung verstanden, die unabhängig vom verwendeten Betriebssystem ist. Die Entwicklung von geeigneter Middleware ist nicht Fokus dieser Arbeit. Es wird vielmehr die Existenz geeigneter Middleware vorausgesetzt, d. h. gewisse Schnittstellen und Anforderungen werden von ihr erfüllt.

Primäre Aufgabe der Middleware ist es, den Kontakt zu anderen Teilnehmern herzustellen, die sich in Kommunikationsreichweite befinden. Ein verbreitetes Verfahren hierzu ist, dass ein ad hoc Knoten in regelmäßigen Zeiteinheiten eine Broad- oder Multicast-Nachricht sendet. In dieser Nachricht sind zumindest sein Identifikator und seine Empfangsadresse enthalten. Wird diese Nachricht von einem anderen Knoten empfangen, ist dieser im Stande eine Nachricht zurückzusenden. Somit ist eine Kommunikation zwischen den ad hoc Knoten hergestellt. Da die Middleware zugleich die direkten Kommunikationspartner verwaltet, wird ein neuer Nachbar als bekannt eingetragen.

Die Broad- oder Multicast-Nachrichten müssen jedoch weiterhin gesendet werden, denn sie können als „Lebenszeichen“ (Lease-Mechanismus) gewertet werden. Bleiben sie aus, hat sich der Knoten entfernt oder wurde deaktiviert und kann aus der Kommunikationspartnerliste entfernt werden.

Zusätzlich ist die Middleware für das Senden und Empfangen von Nachrichten zuständig. Beabsichtigt das MoPiDiG Framework eine Nachricht zu senden, so wird dieses Gesuch an die Middleware weitergeleitet, die anschließend die Nachricht über die Funkschnittstelle versendet. Die Middleware muss jedoch nicht für die Nachrichtenzustellung garantieren, d. h. das MoPiDiG Framework kann nicht davon ausgehen, dass eine versendete Nachricht ihren Empfänger erreicht.

Empfängt die Middleware von einem Knoten eine Nachricht, so muss die Middleware diese an die betreffende MoPiDiG Komponente ausliefern.

Ob und inwiefern die verwendete Middleware in der Lage sein muss, eine Multi-Hop-Kommunikation durchzuführen, hängt von der Applikation ab.

Die Middleware muss außerdem auf mobilen Endgeräten einsetzbar sein, d. h. sie muss mindestens auf PDAs lauffähig sein. Zielplattform sind Mobiltelefone, die momentan auf Grund ihrer Hardwareeinschränkungen nur sehr begrenzt unterstützt werden.

Eine intensive Untersuchung existierender Middleware und eine Bewertung hinsichtlich der Anwendbarkeit für MoPiDiG erfolgt in Kapitel 5.

4.3 Algorithmen im MoPiDiG Framework

Die Algorithmen in MoPiDiG sind der Kern des Frameworks. Es gibt jedoch nicht nur Algorithmen zur Gruppenbildung, sondern wie in Abbildung 4.2 zu sehen ist, auch Al-

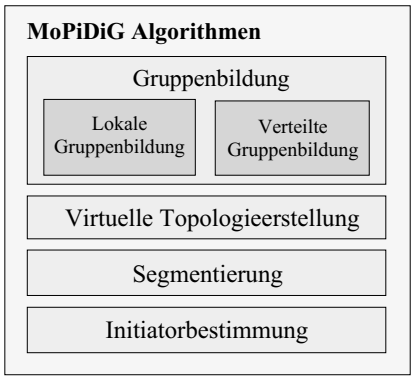


Abbildung 4.2: Aufbau der Algorithmus-Komponente in MoPiDiG

gorithmen zur Initiatorbestimmung, Segmenterzeugung und Topologieerzeugung, die in einer Art Schichtenmodell angeordnet sind.

Die oberste Schicht der Algorithmenarchitektur gemäß Abbildung 4.2 ist für die Gruppenbildung verantwortlich. Im MoPiDiG Framework ist die Gruppenbildung in einen lokalen und einen verteilten Gruppierungsprozess aufgeteilt. Bei der lokalen Gruppierung sucht ein Knoten nur nach Gruppenmitgliedern aus seinen direkten Nachbarn. Es wird hiermit eine Vorselektion erreicht, die als *lokale Gruppe* bezeichnet wird. Im zweiten Schritt werden diese lokalen Gruppen zu allen anderen Knoten gesendet, um so eine oder mehrere *globale Gruppen* zu finden.

Die zwei nächsten Ebenen, die Topologieerstellungs- und Segmentierungsebene, müssen zusammen betrachtet werden. Ad hoc Netzwerke können eine große Anzahl an Teilnehmern enthalten. In solchen Fällen würde der lokale Gruppierungsprozess zu viel Zeit in Anspruch nehmen. Aus diesem Grund ist ein Mechanismus nötig, der ein großes ad hoc Netzwerk in mehrere kleinere Netze aufteilt. Diese Aufgabe übernimmt die Segmentierungsebene.

Die virtuelle Topologieerstellung ist eine Effizienzmaßnahme. In ad hoc Netzwerken ist in vielen Fällen ein vollvermaschtes Netzwerk vorhanden. Durch die Einführung einer einfacher strukturierten virtuellen Topologie - auch als Overlay Netz bezeichnet - kann die Anzahl an Verbindungen reduziert werden. Für die Erstellung einer globalen Gruppe werden folglich weniger Nachrichten benötigt und eine Einigung ist schneller erzielt. Da jedoch die Topologie aufrecht erhalten werden muss, ist die Topologieerstellung nur bis zu einer Geschwindigkeitsobergrenze durchführbar. Deshalb kann auf diesen Vorgang auch verzichtet werden. Nichtsdestotrotz kommt der Topologieerstellung für den Fall von niedrigen Geschwindigkeitsdifferenzen der Teilnehmer eine wichtige Funktion zu.

Die Segmentierung muss von einem Segmentknoten initiiert werden. Damit dieser Vorgang nicht von allen Knoten übernommen wird, wird in der untersten Schicht eine Teilmenge der Knoten bestimmt, die berechtigt ist, die Initiierung zu übernehmen. Dieser Schritt

wird als Initiatorbestimmung bezeichnet. Würden alle Knoten die Initiierung übernehmen, besteht die Möglichkeit, dass keine Segmentierung zu Stande kommt oder sich dies zu einem langwierigen Prozess entwickelt.

Hinsichtlich des Gruppierungsablaufes erfolgt zuerst eine Initiatorbestimmung. Die Segmentierung und Topologieerstellung gehen quasi-parallel vor sich. Abschließend erfolgt die Gruppenbildung, wobei zuerst eine lokale Gruppenbildung und danach die globale Gruppenbildung erfolgt. In Abbildung 4.3 ist der Ablauf an einem Sequenzdiagramm verdeutlicht.

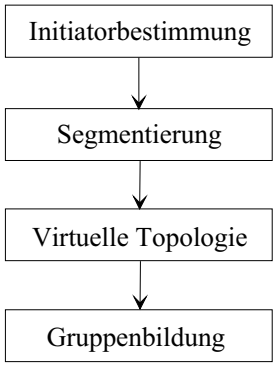


Abbildung 4.3: Ablauf des kompletten Gruppierungsprozesses

Die folgenden Abschnitte beschreiben - beginnend mit der Initiatorbestimmung - die Algorithmen im MoPiDiG Framework.

4.3.1 Initiatorbestimmung

Die Initiatorbestimmung ist ein optionaler Vorgang, der nur benötigt wird, wenn sich ein ad hoc Netzwerk nicht allmählich sondern spontan aufbaut. So könnten nach einer Flugzeuglandung hunderte Personen ihr mobiles Gerät quasi-isochron einschalten. Um in diesen Fällen eine Segmentierung zu starten, wird ein Initiator benötigt.

Auf den ersten Blick könnte die Bestimmung eines Initiators einen Widerspruch gegen die Aufgabenstellung darstellen, in der die Dezentralitätsforderung (siehe Abschnitt 1.1) mit zum Alleinstellungsmerkmal von MoPiDiG beiträgt. Da der Initiator jedoch keine Daten für die Gruppenbildung verwaltet, keine Entscheidungen trifft und auch keine Delegationen vornimmt, kann der Widerspruch ausgeräumt werden.

In der Literatur werden zwei Kategorien von Verfahren diskutiert, wie Initiatoren bestimmt werden können. Aktive Initiatorbestimmung erfordert das Senden von Nachrichten der Teilnehmer, während bei der passiven Initiatorbestimmung implizite Vergleichsmethoden verwendet werden, um aus einer Menge von Teilnehmern einen Ausgezeichneten zu bestimmen.

Nach diesen Ausführungen wird im Folgenden die passive Initiatorbestimmung näher erläutert.

4.3.1.1 Aktive Initiatorbestimmung

Zur aktiven Initiatorbestimmung werden primär Wahlalgorithmen[1] herangezogen. Mattern [Mat89] beschreibt Wahlalgorithmen für verschiedene Topologien (Netz, Baum, Ring etc.). Da die Netzwerke für diese Verfahren statisch sein müssen, ist deren Einsatz für ad hoc Netzwerke nur beschränkt möglich.

Abu-Amara und Lokre [AAL94] untersuchen das Election-Problem in Netzen, in denen häufig Verbindungsfehler auftreten. Ein Designparameter f legt fest, wie viele Verbindungsfehler auftreten dürfen. Als Topologie dient den Autoren jedoch ein vollvermaschtes Netzwerk, so dass einzelne Fehler relativ unkritisch sind, da ein Teilnehmer trotz der Verbindungsfehler über Umwege erreicht werden kann.

Garcia-Molina [GM82] beschreibt den Bully Algorithmus zur Initiatorbestimmung. Beim Bully-Algorithmus hat jeder Knoten i eine a priori zugewiesene Priorität[2] P_i. In einem Netzwerk wird derjenige Knoten Initiator, der die höchste Priorität aufweist. Der Initiator wird dadurch ausfindig gemacht, dass die Prioritäten durch das Netzwerk propagiert werden. Ist der Knoten mit der höchsten Priorität gefunden, informiert dieser sämtliche restlichen Teilnehmer. Der Bully-Algorithmus liefert genau einen Initiator, wenn keine Kommunikationsfehler auftreten.

Beim Auftreten von Kommunikationsfehlern können nach Garcia-Molina [GM82] in zeitlichen Abständen Reorganisationen durchgeführt werden, um auf diese Weise neu hinzugekommene Knoten dem Initiator mitzuteilen.

Hatzis *et al.* [HPS+99] beschreiben Wahlalgorithmen in ad hoc Netzwerken mit und ohne Basisstation. Sie setzen jedoch voraus, dass sich die Anzahl der am Algorithmus teilnehmenden Geräte während der Laufzeit nicht ändert. Ferner ist für den Algorithmus noch ein Mechanismus nötig, der die Streckenlänge misst, die sich durch die Bewegung von Geräten ergibt. Die erste Voraussetzung ist mit ad hoc Netzwerken nur schwer vereinbar, da neue Geräte sowohl zu jeder Zeit hinzukommen als auch verschwinden können. Der zweite Punkt widerspricht der Anforderung aus der Aufgabenstellung in Abschnitt 1.1, nachdem eine Ortskoordinate nicht Voraussetzung für das MoPiDiG Framework ist.

Malpani *et al.* [MWV00] adaptieren das Routing-Protokoll TORA ([PC97]) und das Verfahren von Gafni und Bertsekas [GB81]. Der durch die Änderungen entstandene *Leader*-Algorithmus ist für ad hoc Netzwerke einsetzbar, da sich die Topologie in mehreren Knoten ändern kann.

Der Vorteil der aktiven Initiatorbestimmung besteht darin, dass als Ergebnis genau ein Initiator festgelegt wird. Um dies zu erreichen, muss jedoch eine Vielzahl an Nachrichten versendet werden. Deshalb ist diese Art der Initiatorbestimmung sehr zeitaufwändig. Ändert sich während der Initiatorbestimmung die Anzahl der Teilnehmer, verlängert sich diese Zeit noch zusätzlich.

Zu beachten ist, dass der Initiator durch das Versenden von Zusatznachrichten eine Mehrarbeit leistet. Deswegen müsste ein Initiator entweder ein Art „Vergütung“ erhalten oder das Verfahren ist fair, so dass die Wahrscheinlichkeit, dass ein Teilnehmer Initiator wird, für alle gleich groß ist. Hinsichtlich der Fairness wird bei den erwähnten Algorithmen jedoch keine Aussage gemacht.

[1] Wahlalgorithmen sind in der Literatur auch als Election- oder Votingalgorithmen bekannt.

[2] Die Priorität entspricht dem im MoPiDiG Framework geforderten Identifikator gemäß Abschnitt 4.1.2

4.3.1.2 Passive Initiatorbestimmung

Obige Ausführungen zu den aktiven Methoden lassen erkennen, dass diese zu komplex für die Anforderungen im MoPiDiG Framework sind.

Die passiven Methoden führen im Gegensatz zu den aktiven i. A. zu mehreren Initiatoren. Dies stellt keinen Nachteil dar. Grund für die Initiatorbestimmung ist die *Reduzierung* der Geräte, die die Segmentierung initiieren. Deswegen ist es nicht Voraussetzung, dass nur ein einziger Initiator existiert.

Bei der passiven Initiatorbestimmung werden nur in Ausnahmefällen Nachrichten versendet. Grundlage für die passive Initiatorbestimmung sind die Identifikatoren, die gemäß der Annahmen in Abschnitt 4.1.2 vorhanden sein müssen. Damit ein Teilnehmer von seiner Initiatorrolle erfährt, werden bestimmte Bedingungen an seinen Identifikator gestellt. Sind diese erfüllt, so ergibt sich die Initiatorrolle implizit.

In einer ersten Variante wird ein Initiator dadurch bestimmt, dass der Identifikator eines Teilnehmers mit denen seiner direkten Nachbarn verglichen wird. Da gemäß Anforderung in Abschnitt 4.1.2 jeder Knoten auch die Identifikatoren seiner direkten Nachbarn kennt, wird ohne Nachrichtenaustausch entschieden, ob ein Knoten Initiator ist oder nicht.

Bei diesem Verfahren ist gesichert, dass im ganzen Netzwerk mindestens ein Initiator existiert - der mit dem niedrigsten Identifikator.

Ist die Inititiatorbestimmungsbedingung von den Nachbarn eines Teilnehmers abhängig, kann sich eine ungleichmäßige Verteilung der Initiatoren in einem Netzwerk ergeben. So fungieren in solchen Fällen primär die Teilnehmer als Initiator, die wenige direkte Nachbarn besitzen. Der Grund hierfür ist, dass die Wahrscheinlichkeit, dass ein Nachbaridentifikator niedriger ist, bei weniger Nachbarn kleiner als bei einer hohen Anzahl an Nachbarn - sofern angenommen wird, dass die Identifikatoren gleich verteilt sind. Betroffen sind vor allem Teilnehmer, die sich im Randbereich des ad hoc Netzwerkes aufhalten, da dort die Nachbaranzahl immer niedriger ist.

Eine Möglichkeit weitgehend unabhängig von Nachbaridentifikatoren zu sein ist eine Initiatorbestimmungsfunktion f_I. Sei $\mathbb{W}_I$ der Wertebereich der Identifikatoren, so ist $f_I : \mathbb{W}_I \mapsto \mathbb{N}$. Die Funktion f_I sollte einfach zu berechnen sein. Es bieten sich hierfür insbesondere Integerarithmetik und Bitoperationen an.

Eine mögliche Funktion wäre die Modulo-Division

$$I \equiv 0 \mod k,$$

die den Rest einer ganzzahligen Division angibt[3]. Ein Teilnehmer wird Initiator, wenn die Modulo Division seines Identifikators I mit einer a priori bekannten Zahl k ein bestimmtes Ergebnis ergibt.

Bei diesem Verfahren kann es allerdings geschehen, dass zwei direkte Nachbarn die Initiatorrolle innehaben. Soll dieser Fall vermieden werden, müssen auch bei der passiven Initiatorbestimmung Nachrichten versendet werden. Hat ein Teilnehmer durch Berechnung festgestellt, dass er Initiator ist, muss er zusätzlich überprüfen, ob dies auch auf einen seiner Nachbarn zutrifft. Steht dies fest, so wird dieser Nachbar mit einer entsprechenden

[3]Sieben modulo drei ist der Rest, der bei der Division von sieben durch drei übrig bleibt, d. h. eins. In obiger Notation: $7 \equiv 1 \mod 3$

Nachricht kontaktiert. Inhalt dieser Nachricht ist ferner eine Zufallszahl. Haben beide diese Zahl ausgetauscht, wird derjenige zum Initiator, der die größere[4] Zahl hat.

Zu bedenken ist außerdem, dass im gesamten Netzwerk kein Knoten existieren muss, der die Initiatorbestimmungsfunktion erfüllt und somit kein Initiator zur Verfügung steht. Gilt für den Initiator mit dem Identifikator I : $I \equiv 0 \mod k$, so ist die Wahrscheinlichkeit p_I, dass bei n Knoten kein Initiator existiert:

$$p_I = \left(1 - \frac{1}{k}\right)^n \Leftrightarrow k \leq \left\lfloor \frac{1}{1 - \sqrt[n]{p_I}} \right\rfloor. \tag{4.1}$$

Mit dem rechten Ausdruck in Gleichung 4.1 kann ein k für eine gewünschte Wahrscheinlichkeit p_I berechnet werden. Für die Wahl von k ist damit die Größe des Netzwerkes wesentlich. Diese Information ist aber i. A. nicht erhältlich und kann nur anwendungsspezifisch geschätzt werden.

Bezüglich einer oberen Grenze kann bei gleichverteilten Identifikatoren von $\frac{n}{k}$ Initiatoren für ein Netzwerk mit n Teilnehmern ausgegangen werden.

Im MoPiDiG Framework werden beide vorgestellten passiven Initiatorbestimmungsmethoden verwendet. Aktive Bestimmungsmethoden finden auf Grund des notwendigen Nachrichtenaustauschs keine Verwendung.

4.3.2 Segmentierung eines ad hoc Netzwerks

Ad hoc Netzwerke sind a priori nicht in ihrer Größe beschränkt. Würde eine Gruppenbildung in einem Netzwerk mit vielen Teilnehmern durchgeführt, so nähme dies zu viel Zeit in Anspruch. Aus diesem Grund wird in MoPiDiG ein ad hoc Netzwerk in mehrere kleine Netzsegmente aufgeteilt, wie in Abbildung 4.4 gezeigt. Die Gruppenbildung wird anschließend nur innerhalb eines solchen Segments durchgeführt.

Nachteilig wirkt sich dieses Vorgehen dadurch aus, dass durch die Aufteilung potenzielle Gruppenmitglieder verloren gehen, bzw. Gruppen nicht gebildet werden können. So kann es geschehen, dass zwei Personen, die Teil verschiedener Segmente sind, eine Gruppe bilden könnten, dies aber durch die Aufteilung in Segmente nicht geschehen kann.

Diese Problematik lässt sich nicht lösen, wird aber durch die Tatsache sukzessive vermindert, dass die Reichweiten der verschiedenen Funktechnologie sich permanent erhöhen. Hierdurch vergrößern sich die Segmente und da als Gruppenteilnehmer nur Personen in der momentanen Umgebung in Frage kommen, wird das Problem darauf zurückgeführt, wie weit ein potenzielles Gruppenmitglied generell für die Gruppenbildung entfernt sein darf.

4.3.2.1 Formale Definition

Bevor die Erzeugung eines Segments im Detail beschrieben wird, folgt eine formale Definition sowie die Beschreibung der Eigenschaften eines Segments und der Segmentierung eines Netzwerkes im Allgemeinen.

[4] Ob in diesem Fall die größere oder die kleinere Zahl „gewinnt“ ist nebensächlich, es muss nur eine der zwei Alternativen gewählt werden.

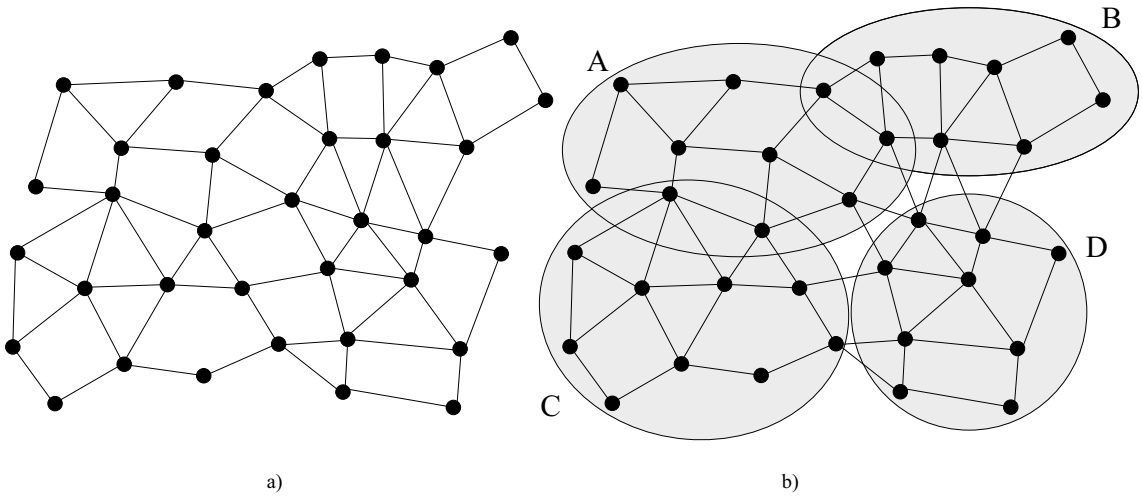


Abbildung 4.4: Segmentierung eines Graphen in Teilgraphen

Eine Definition des Segmentbegriffs erfolgte bereits in Abschnitt 3.1.5. An dieser Stelle soll jedoch noch eine Verbindung zu dem Graph und den entstehenden Segmenten hergestellt werden. So sei im Folgenden mit $\mathcal{S} = \{S_1, S_2, \ldots S_n\}$ die Menge der Segmente eines Graphen $G = (V, E)$ bezeichnet, d. h. $S_i \subseteq G$.

Ein Graph (d. h. Netzwerk) soll gänzlich in Segmente aufgeteilt werden, d. h. $\forall v \in V : \exists S_i = (V_{S_i}, E_{S_i}) \in \mathcal{S}$, so dass $v \in V_{S_i}$. Die Existenz einzelner, nicht in Segmenten enthaltener Knoten ist unerwünscht, da sie nicht an Gruppierungen teilnehmen können.

Die Segmente eines ad hoc Netzwerks in $\mathcal{S}$ müssen für das MoPiDiG Framework nicht disjunkt sein, d. h. für $S_i = (V_{S_i}, E_{S_i}) \in \mathcal{S}$ und $S_j = (V_{S_j}, E_{S_j}) \in \mathcal{S}$ darf gelten: $V_{S_i} \cap V_{S_j} \neq \varnothing$. Die Forderung nach disjunkten Segmenten wäre zu strikt. Wenn die Randknoten eines Segments noch anderen Segmenten angehören, so ist dies durchaus wünschenswert. Randknoten sind auf Grund ihrer Lage i. A. die ersten Knoten, die das Segment wieder verlassen. Eine Bewegung führt in solchen Fällen nicht zu segmentlosen Knoten, da diese bereits in anderen Segmenten integriert sind. Knoten, welche nicht Randknoten sind, sollten jedoch nicht in mehreren Segmenten auftreten.

Bei nicht disjunkten Segmenten gehört ein Knoten und damit sein Teilnehmer mindestens zwei Segmenten an. Damit könnte der Teilnehmer auch an zwei Gruppierungen gleichzeitig teilnehmen. Hiermit tritt allerdings das Problem auf, dass der Teilnehmer nach der Gruppenbildung nur in einer Gruppe präsent sein kann. Der Teilnehmer müsste sich am Ende der beiden Gruppierungen entscheiden, welche Gruppe für ihn mehr Nutzen bringt. Für die andere Gruppe würde das bedeuten, dass jeder Teilnehmer über die Absenz benachrichtigt werden muss, was durchaus noch ein vertretbarer Schritt wäre. Doch wesentlich mehr zum Tragen kommt die potenziell nicht vorhandene Optimalität der nicht gewählten Gruppe. Dadurch dass ein Teilnehmer fehlt, könnte eine völlig andere Gruppenkonstellation möglich sein. Um sicher zu gehen wäre eine Regruppierung notwendig, die aus zeitlichen Gründen meist nicht möglich ist. Um festzustellen, ob zwei Gruppie-

rungen voneinander abhängen, wäre zudem ein entsprechender Erkennungsmechanismus notwendig, der zum jetzigen Zeitpunkt noch nicht Teil von MoPiDiG ist.

Ist in der vorliegenden Implementierung ein Knoten Element mehrerer Segmente, muss dieser sich entscheiden, in welchem Segment er sich der Gruppierung anschließt. Die Entscheidung wird zu einem Zeitpunkt gefällt, zu dem noch keine globale Gruppenbildung gestartet wurde.

4.3.2.2 Erzeugung von Segmenten

Im Folgenden wird die spontane und graduelle Erzeugung von Segmenten erklärt. Die spontane Erzeugung wird angewandt, wenn das Netzwerk bereits aufgebaut ist. Bei der graduellen Methode bilden sich Netzwerk und Segmente parallel.

Spontane Segmenterzeugung: Unter der spontanen Erzeugung der Segmente wird der Vorgang verstanden, bei dem mit Hilfe der Initiatoren, die im vorherigen Abschnitt bestimmt wurden, ein ad hoc Netzwerk in Segmente aufgeteilt wird.

Der Algorithmus orientiert sich an den Arbeiten von Gerla und Tsai [GT95] und Chen und Stojmenovic [CS99], wobei im Falle von MoPiDiG primär disjunkte Segmente erzeugt werden, in denen eine eine k-hop Kommunikation möglich sein soll (siehe Tabelle 3.2).

Die prinzipielle Idee des Verfahrens besteht darin, dass ausgehend von einem Initiator sternförmig neue Knoten in das Segment mit aufgenommen werden, sofern diese noch nicht Teil eines anderen Segments sind.

Um die Größe des Segments festzulegen, ist ein Parameter notwendig. Die Anzahl an Knoten im Segment als Designparameter scheidet leider aus, weil diese in einem verteilten System a priori nicht bekannt ist. Es ist durchaus möglich Segmente mit einer bestimmter Größe zu erzeugen, die notwendigen Verfahren hierzu beruhen jedoch meist auf zentralen Komponenten, die die Größe des Netzwerksegments überwachen. Die verteilten Verfahren erfordern hohen Synchronisationsaufwand.

Als Designparameter für ein Segment dient der halbe Durchmesser $\mathcal{R}_G$ eines Graphen. Als Durchmesser eines Graphen wird der größte Abstand zweier Knoten in G definiert, wobei als Abstand die geringste Länge zweier einzelnen Knoten bezeichnet wird [Die00]. Zur Realisierung dient ein Hop-Counter. Die Anzahl an Sprüngen (Hops) entspricht dem halben Durchmesser.

Ausgangspunkt für die Segmentierung sind die Initiatoren, die in Abschnitt 4.3.1 bestimmt wurden. Zusätzlich ist noch die Definition des Durchmessers des Segments nötig. Dieser wird nicht als einzelner Wert, sondern als Intervall $I_D = [k, k+1]$ $(k \in \mathbb{N}_0)$ angegeben. k wird abhängig vom Geschwindigkeitsverhalten gewählt.

Von den Initiatoren ausgehend wird im ersten Schritt eine `SEGMENT_CREATION` Nachricht an alle direkten Nachbarn gesendet. In der Nachricht muss das Intervall I_D enthalten sein. Bei Empfang einer Nachricht werden zuerst beide Intervallwerte dekrementiert, gespeichert und wenn der untere Schwellwert größer als null ist, wird die `SEGMENT_CREATION` Nachricht mit angepassten I_D weitergeleitet. Empfängt ein Knoten mehrere `SEGMENT_CREATION` Nachrichten, so werden nur beim Erhalt der ersten Nachricht die Intervallgrenzen erniedrigt und die Nachricht weitergeleitet. Der Pseudocode ist in Abbildung 4.5 zu sehen.

Abbildung 4.6a zeigt die Segmentierung eines Netzwerks nach dem ersten Schritt. Die

```
Initiatorknoten:
    initialisiere Intervall I_D=[k, k+1];
    sende SEGMENT_CREATION Nachricht mit I_D an alle Nachbarn;

nicht-Initiatorknoten:
    init:
        creation_msg_received = false;

    Bei Empfang einer SEGMENT_CREATION Nachricht:
        if( !creation_msg_received )
            creation_msg_received = true;
            decrement Intervall I_D;
            send SEGMENT_CREATION Nachricht mit I_D an alle Nachbarn;
        fi;
```

Abbildung 4.5: Pseudocode für Segmenterzeugung

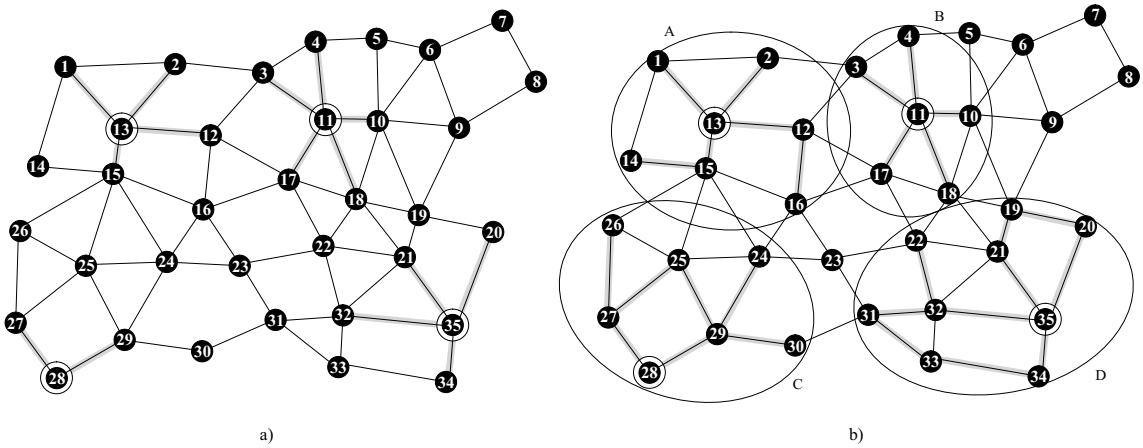


Abbildung 4.6: Ablauf der Segmentierung eines Netzwerks

umrandeten Knoten mit den Nummern 11, 13, 28 und 35 fungieren als Initiator und das Intervall I_D beträgt $I_D = [1, 2]$.

Bei dem beschriebenen Verfahren kann es geschehen, dass Elemente des Netzwerkes keinem Segment angehören, weil die Hop-Distanz zu diesen Elementen zu groß ist, siehe z. B. Knoten 23 in Abbildung 4.6b. Es kann sich um einzelne Punkte handeln, die noch in das Segment mit aufgenommen werden könnten oder um eine größere Anhäufung, die ein eigenes Segment bilden könnte, aber mangels Initiator (kann nur in Variante 2 der passiven Initiatorbestimmung geschehen, siehe Abschnitt 4.3.1.2) nicht dazu in der Lage war.

Nach dem ersten Schritt hat das Segment einen Durchmesser von $2k$. Die Randpunkte senden an ihre Nachbarn eine `SEGMENT_LOWER_LIMIT_ACHIEVED` Nachricht mit beigefügter

I_D. Empfängt ein Knoten A solch eine Nachricht, hängt sein Verhalten von diversen Faktoren ab.

Ist A bereits Teil eines anderen Segments, so sendet er eine `ALREADY_SEGMENT_MEMBER` Nachricht zurück und die Segmentierung ist erledigt. Hat Knoten A keinen anderen Nachbarn als den Randpunkt, so sendet A eine `SEGMENT_ADD_SINGLE_REQUEST`-Nachricht zum Randpunkt. Existiert dieser Randpunkt noch, wird er i. A. mit `SEGMENT_ADD_REQUEST_AGREE` antworten und A ist Teil des Segments.

Im letzten Fall hat Knoten A neben dem Segmentrandpunkt noch andere Nachbarn. Bei diesen gilt es mit Hilfe einer `SEGMENT_MEMBER_QUERY` Nachricht anzufragen, ob sie schon Mitglied in einem anderen Segment sind. Diese antworten gemäß ihrem Zustand mit `ALREADY_SEGMENT_MEMBER` oder `ALREADY_NOT_SEGMENT_MEMBER` Nachrichten. Ist jeder Nachbarknoten Teil eines Segments wird wie im vorherigen Fall mit einer `SEGMENT_ADD_SINGLE_REQUEST`-Nachricht um Aufnahme in das Segment erbeten. Ist mindestens ein Nachbar kein Segmentmitglied, wird Knoten A zum Initiator eines neuen Segments bestimmt. Das neue Segment wird wie im ersten Schritt erzeugt mit der Ausnahme, dass das Intervall mit $I_D = [2k, 2k + 1]$ initialisiert wird. `SEGMENT_CREATION`-Nachrichten werden nur an die Nachbarn gesendet, deren Segmentanfrage negativ war. Zusätzlich wird der Randpunkt, der Knoten A hätte aufnehmen sollen, durch eine `SEGMENT_CREATION` Teil des neuen Segments. Abbildung 4.7 veranschaulich den kompletten Prozess in einem Sequenzdiagramm.

Wie in Abbildung 4.6b zu sehen ist, bleibt Knoten 23 ohne direkten Segmentkontakt. Sämtliche angrenzenden Segmente haben ihre maximale Größe bereits erreicht. Folglich erzeugt Knoten 23 ein eigenes Segment, welches aus den angrenzenden Segmentrandpunkten (die Knoten 16, 22, 24, 31) besteht.

Die Knoten 5-9 sind ebenfalls noch nicht Mitglied eines Segments. Wenn Knoten 10 dem Knoten 6 mit `SEGMENT_LOWER_LIMIT_ACHIEVED` mitteilt, dass die untere Segmentgrenze erreicht ist und Knoten 6 daraufhin feststellt, dass er noch weitere unsegmentierte Knoten als Nachbarn besitzt, wird Knoten 6 nicht in das Segment B aufgenommen. Knoten 6 wird deswegen Initiator eines neuen Segments. Da Knoten 19 aus Segment D und Knoten 9 ebenso verfahren, wird auch Knoten 9 zu einem Initiator. Da zwei benachbarte Initiatoren nicht erwünscht sind, führt nur eine Initiationsrunde zum Erfolg.

Der Segmentierungsalgorithmus benötigt k Schritte. Um sämtliche Knoten zu erreichen sind mindestens so viele Nachrichten notwendig, wie es Segmentknoten n_S gibt. Dies ist jedoch nur der Fall, wenn eine Sterntopologie vorliegt. Ist mit e_S die Anzahl an Kanten im Segment angegeben, beträgt die obere Schranke $2e_S$ Nachrichten, da ein Knoten die Nachricht an all seine Nachbarn, bis auf den Sender weiterleitet. Jeder Knoten leitet die Segmenterzeugungsnachricht jedoch nur einmal weiter, so dass über jede Kante maximal zwei Nachrichten gesendet werden, was zu der Schranke von $2e_S$. Somit ergibt sich die Komplexitätsordnung von $\mathcal{O}(e_S)$, bzw. $\mathcal{O}(n_S^2)$ wenn die maximale Vernetzung von $e_S^{max} = \frac{n(n-1)}{2}$ angenommen wird.

Bezüglich der Korrektheit des Algorithmus kann sofort angegeben werden, dass er in jedem Fall terminiert, da der Algorithmus den Parameter k bei jedem Schritt dekrementiert und bei einem k-Wert von Null der Algorithmus stoppt.

Ferner erzeugt der Algorithmus Segmente, da ausgehend vom Initiatorknoten Knoten mit Hop-Distanz k logisch zusammengefasst werden. Die entstandenen Segmente müssen

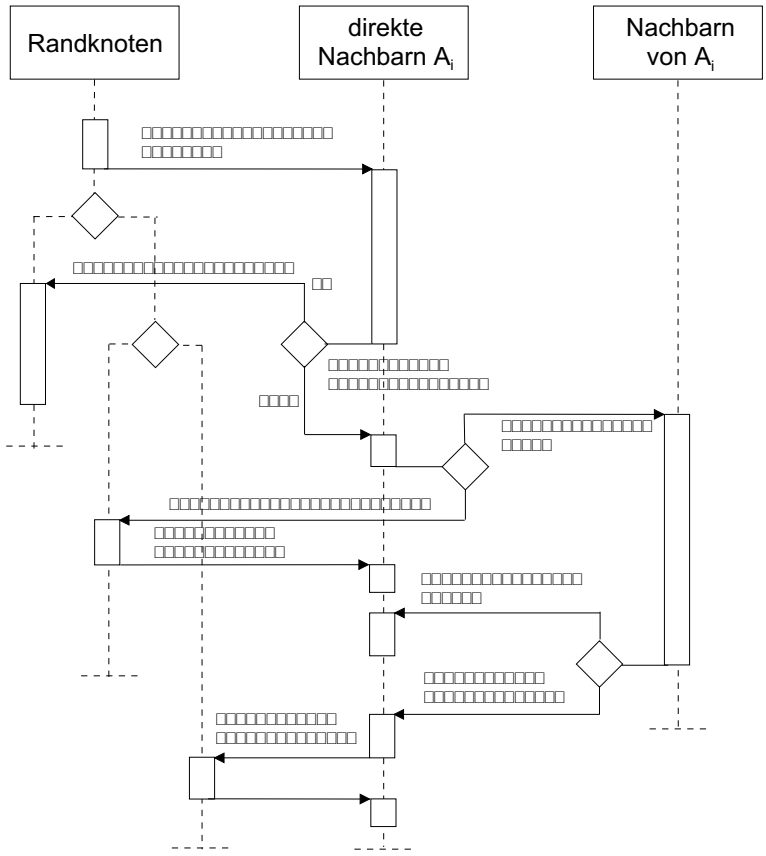


Abbildung 4.7: AUML-Sequenzdiagramm zur Verdeutlichung der Addition von Segmentteilnehmern

jedoch nicht disjunkt sein. Dies ist jedoch kein Resultat des Segmenterzeugungsalgorithmus sondern die Ursache ist in der Verteilung der Initiatoren zu suchen.

Graduelle Segmenterzeugung: Ein Segment kann sich aber auch während eines Gruppierungsvorganges allmählich aufbauen. So können mehrere Teilnehmer eine Gruppierung initiieren, wobei sie sich aber noch nicht alle innerhalb ihrer Kommunikationsreichweite befinden. Erst schrittweise nähern sich Knoten einander an und bilden ein Segment. Ein Initiator ist nicht vorhanden. Da die Middleware die zu einem Zeitpunkt möglichen direkten Kommunikationspartner kennt, erfährt das MoPiDiG Framework ebenfalls, ob neue Knoten hinzukommen.

Das Erzeugen eines Segments in diesem Fall geschieht prinzipiell wie das Hinzufügen neuer Teilnehmer im oben beschriebenen Verfahren. Begegnen sich zwei Knoten, die nicht Teil eines Segments sind, so bilden diese beiden die Grundlage eines neuen Segments. Bezüglich der Initialisierung der Knoten mit dem Intervall I_D, wird festgelegt, dass der Knoten mit dem höheren Identifikator mit $I_D = [k, k+1]$ und der verbliebene mit $I_D = [k-1, k+1]$ initialisiert wird.

Will ein singulärer Knoten sich dem Segment anschließen, sendet dieser eine `SEGMENT_ADD_SINGLE_REQUEST`-Nachricht an den Randknoten des Segments. Dieser Randknoten handelt abhängig vom Intervallwert. Ist die obere Schranke noch nicht erreicht, so wird eine `SEGMENT_ADD_REQUEST_AGREE` im anderen Fall eine `SEGMENT_ADD_REQUEST_REFUSE` Nachricht gesendet. Im letzten Fall wird der singuläre Punkt selbst zu einem Segment. Erste Mitglieder im Segment sind die Randpunkte des anderen Segments, zu denen er Kontakt hat.

Das Hinzufügen eines Knotens ist von konstanter Komplexität, d. h. $\mathcal{O}(1)$, da stets die gleiche Anzahl an Schritte notwendig ist. Diese besteht aus dem Aussenden einer Anfrage und das Zurücksenden der Antwort.

Entfernt sich ein Randknoten ist das Aussenden von Nachrichten nicht notwendig, nur der Parameter k muss dekrementiert werden. Die Komplexität der Entfernung eines Nicht-Randknotens entspricht dem Fall der spontanen Segmenterzeugung.

Es kann auch geschehen, dass sich zwei Segmente einander nähern, d. h. es kann ein Knoten in die Reichweite eines Segments kommen, welcher bereits Teil eines anderen Segments ist. Beide Segmente können verschmolzen werden, wenn hierdurch der Gesamtdurchmesser des Segments nicht überschritten wird. Ob die Verschmelzung möglich ist, wird durch das Senden einer `SEGMENT_ADD_MULTIPLE_REQUEST` Nachricht geprüft. In der Nachricht muss die Größe des anzufügenden Segments enthalten sein. Ist ausreichend Kapazität vorhanden beide Segmente zu verschmelzen, wird dies mit einer `SEGMENT_ADD_REQUEST_AGREE` Nachricht bestätigt. Anschließend müssen die Intervallschranken in einem Segment (in der vorliegenden Implementierung das Segment, welches die Request-Nachricht gesendet hat) angepasst werden. Ansonsten wird mit einer `SEGMENT_ADD_REQUEST_REFUSE` Nachricht mitgeteilt, dass beide Segmente getrennt bleiben. Abbildung 4.8 zeigt den Ab-

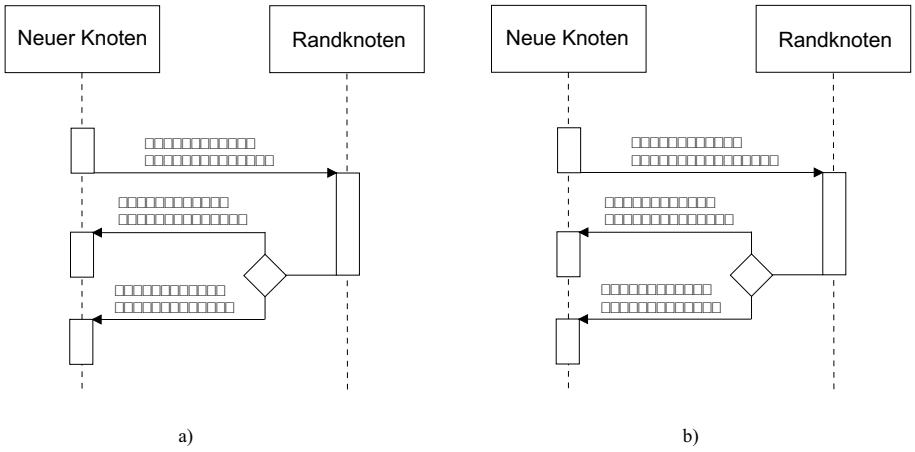


Abbildung 4.8: Sequenzdiagramm zur graduellen Segmenterzeugung

lauf in einem Sequenzdiagramm.

In einem Spezialfall dürften Knoten noch hinzugefügt werden, obwohl die Intervallobergrenze bereits erfüllt ist. Ein Beispiel ist in Abbildung 4.9 dargestellt. Abbildung 4.9a zeigt das Beispielnetzwerk. In Abbildung 4.9b ist zu sehen, dass sich der Durchmesser

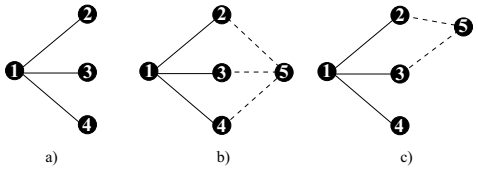


Abbildung 4.9: Segmentänderung - Hinzufügen von Knoten

des Graphen durch die Addition des Punktes Nummer 5 nicht ändert. Der Durchmesser hat vor und nach der Addition den Wert zwei. Erfüllt ist dies jedoch nur, wenn sämtliche Randknoten mit dem neuen Knoten verbunden werden und somit Ringstrukturen entstehen. In Abbildung 4.9c ist diese Bedingung nicht erfüllt und dadurch erhöht sich der Durchmesser auf den Wert drei.

Da dieser Ausnahmefall für beliebige k-Werte algorithmisch nur sehr aufwändig zu lösen ist und der Spezialfall - wenn überhaupt - nur in kleinen Segmenten auftritt, wird auf eine Betrachtung hier verzichtet.

Segmentidentifikation: Mit jedem Segment ist ein Segmentidentifikator verbunden. Diese ID entspricht dem Identifikator des Initiators. Da ein Knoten Element mehrerer Segmente sein kann, ist der Segmentidentifikator notwendig um die Segmentteilnehmer auf die Segmente abzubilden. So verwaltet jeder Knoten eine Tabelle an Segmentidentifikatoren, die je mit einer Liste von Ausgangsknoten assoziiert sind. Werden Knoten dem Segment hinzugefügt, so muss diesen der Segmentidentifikator mitgeteilt werden.

Mit Hilfe des Segmentidentifikators ist es ferner möglich festzustellen, ob ein neu hinzukommender Knoten bereits Mitglied des Segments war. Grundlage hierfür ist, dass sich jeder Knoten die letzten Segmente, in denen er Mitglied war, merkt. Welche Bedeutung den Segmentidentifikatoren beim Gruppierungsprozess zukommt, wird in Abschnitt 4.3.4.2 erläutert.

4.3.2.3 Aufrechterhaltung der Segmente

Neben der Erzeugung der Segmente ist deren Aufrechterhaltung von wesentlicher Bedeutung. Da sich Knoten aus dem Segment entfernen können, ändert sich der Durchmesser des Segments, so dass an anderer Stelle neue Knoten hinzugefügt werden können. Abbildung 4.10 verdeutlicht diesen Sachverhalt. In Abbildung 4.10a ist ein Segment mit Durchmesser von drei Kanten dargestellt. Keiner der Knoten 1 oder 4 dürfte neu hinzukommende Knoten aufnehmen um die Durchmesserschranke nicht zu überschreiten. In Abbildung 4.10b verschwindet der Knoten 1 und ein neuer Knoten 5 kommt hinzu. Durch das Verschwinden von Knoten 1 hat sich der Durchmesser des Segments erniedrigt, so dass der Knoten 5 in das Segment aufgenommen werden darf. Abbildung 4.10c zeigt das neu entstandene Segment.

Bevor ein Verfahren zur Aufrechterhaltung vorgestellt wird, muss noch deren zeitlicher Rahmen und deren Notwendigkeit diskutiert werden. Die Aufrechterhaltung ist während der Gruppenbildung notwendig, bis ein endgültiges Ergebnis feststeht. In diesem Zeitraum

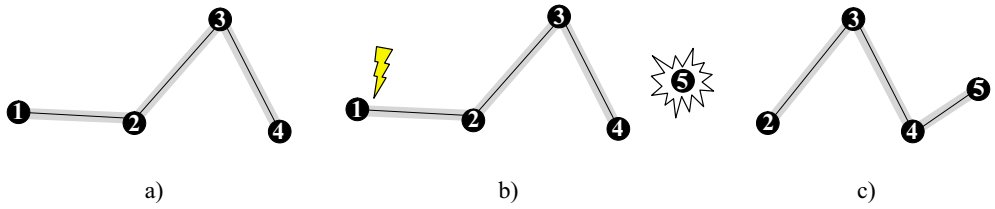


Abbildung 4.10: Segmentänderung bei Knotenreduktion

müssen neu hinzugekommene Knoten die Möglichkeit bekommen sich an der Gruppierung zu beteiligen. Eine Aufrechterhaltung der Segmentierung zwischen zwei Gruppierungen bietet sich i. A. nicht an. Es empfiehlt sich vor jeder Gruppierung die Segmente neu zu erstellen, da Gruppierungsvorgänge nicht ständig stattfinden und eine Aufrechterhaltung der Segmente während dieser Zeit zu ressourcenintensiv ist, d. h. es wäre ein hoher Kommunikationsaufwand notwendig.

Ein Teil der Aufrechterhaltung wurde bereits diskutiert. Die Hinzunahme von singulären Knoten oder anderen Segmenten bei der allmählichen Erzeugung entspricht dem Vorgehen bei der Aufrechterhaltung des Segments. Was bisher unerwähnt blieb, ist das Verschwinden von Knoten aus dem Segment und die damit verbundene Aktualisierung des Intervalls I_D.

Wird von einem Knoten A entdeckt, dass zu einem Knoten B kein Kontakt mehr besteht, überprüft Knoten A, wie viele Kommunikationspartner er noch hat. Hat Knoten A nur eine ausgehende Kante, so sendet er eine `SEGMENT_UPDATE` Nachricht an seinen Nachbarn, die durch das ganze Segment weitergeleitet wird. Auf diese Nachricht reagiert jeder Knoten so, dass er seine Intervalluntergrenze inkrementiert. Die Nachricht muss mit einer `UPDATE_ID` versehen sein, da ein Knoten diese Nachricht mehr als einmal zugestellt bekommen kann, aber der Intervallwert nur einmal inkrementiert werden darf.

Kann ein Knoten noch mit mehr als einem Knoten kommunizieren, so muss der Durchmesser des Segments neu bestimmt werden, da sich mit dem entfernten Knoten B noch weitere Knoten entfernt haben können. Hierzu muss das Intervall I_D für jeden Knoten neu bestimmt werden, was der Knoten A initiiert.

4.3.3 Virtuelle Topologie

Nachdem ein Verfahren vorgestellt wurde, ein großes ad hoc Netzwerk in mehrere Segmente zu zerlegen, steht in diesem Abschnitt die Überlagerung des Segments mit einer virtuellen Topologie[5] im Vordergrund. Bevor eine virtuelle Topologie zum Einsatz kommt, muss überprüft werden, ob die Geschwindigkeitseigenschaften einer Anwendung dies erlauben, da sich der Mehraufwand, der durch die Topologieerzeugung erzeugt wird, nur bis zu einer gewissen Differenzgeschwindigkeit amortisiert. Jenseits dieser oberen Schranke ist die Anwendung einer virtuellen Topologie nicht mehr als nutzbringend anzusehen. Details über die Anwendbarkeit der virtuelle Topologie und damit verbundenen oberen Schranken

[5]im Zusammenhang mit Peer-2-Peer Netzen auch oft als Overlay-Technik bezeichnet.

sind in Kapitel 6 zu finden.

Für die Gruppenbildung muss eine Nachricht, welche von einem Knoten ausgeht, an alle restlichen Knoten gesendet werden um deren Profilinformation zu erhalten. Eine virtuelle Topologie wird erstellt um weniger Nachrichten senden zu müssen, wenn jeder Knoten eines Segments erreicht werden soll. Zu sehen ist das in Abbildung 4.11. Ohne zusätzliche To-

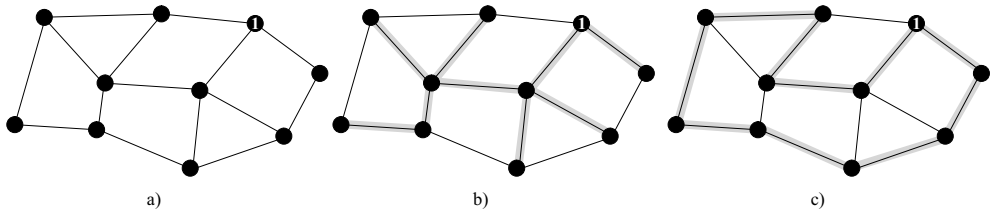


Abbildung 4.11: Verwendung einer virtuellen Topologie. a) zeigt das ad hoc Netzwerk. In b) wird das ad hoc Netzwerk mit einer Baumtopologie überlagert, während in c) ein Ring als virtuelle Topologie verwendet wird.

pologie sind in Abbildung 4.11a 15 Kanten vorhanden. Durch die Überlagerung mit einem Baum (Abbildung 4.11b) reduziert sich dies auf neun Kanten. Für einen Ring sind zehn Kanten nötig. Konkret bedeutet dies, dass in Abbildung 4.11a Knoten 1 drei direkte Nachbarn besitzt und an diese Nachrichten sendet. In Abbildung 4.11b und Abbildung 4.11c hat Knoten 1 jedoch nur zwei Nachbarn. Durch die Reduktion der Nachrichtenanzahl erfolgt die Verteilung der Nachrichten schneller. Werden ferner weniger Nachrichten gesendet, so müssen auch weniger empfangen werden, woraus ein geringerer Energieverbrauch resultiert.

4.3.3.1 Formale Definition

Bevor detailliert die Erzeugung von verschiedenen Topologien erläutert wird, erfolgt eine formale Definition einer virtuellen Topologie.

Definition 5: *Virtuelle Topologie*
Sei $S = (V_S, E_S)$ ein Segment, dann ist eine virtuelle Topologie $T_V = (V_S, E_{T_V})$ ein Teilgraph von S mit $E_{T_V} \subseteq E_S$. Ferner ist $\forall v_i, v_j \in V_S : \exists$ ein Pfad $P = \{e_1, \ldots, e_n\} = \{(v_i, v_{i+1}), (v_{i+1}, v_{i+2}), \ldots, (v_{j-1}, v_j)\}$, wobei gilt $\forall e_i \in P : e_i \in E_{T_V}$.

□

Eine virtuelle Topologie besteht somit aus sämtlichen Knoten eines Segments, aber nur aus einer Teilmenge der Kanten. Es muss zusätzlich für je zwei Knoten aus der Knotenmenge V_S eine Verbindung von Kanten bestehen, die alle Element aus E_{T_V} sind.

Als virtuelle Topologien kommen viele Strukturen in Frage. Primäres Auswahlkriterium ist deren effiziente Erstellbarkeit. Ferner soll die virtuelle Topologie *robust* sein, d. h. durch die Bewegung der Teilnehmer sollten nur wenig Änderungen provoziert werden. Änderungen in der Topologie sollen mit möglichst wenig Aufwand korrigiert werden können.

Aus diesen Gründen werden in diesem Abschnitt als virtuelle Topologien die Baum- und Ringtopologie erklärt und bewertet. Alternativen hierzu, wie z. B. Sterntopologie, Gittertopologie oder Hyperkubus, bleiben unberücksichtigt, da entweder deren Erzeugung zu aufwändig ist oder die Topologie Anforderungen benötigt, die von einem ad hoc Netzwerk nicht generell erfüllt werden können. So muss beispielsweise in einem Gitter ein Knoten mit einer bestimmten Mindestanzahl an anderen Knoten verbunden sein, um die Gitterstruktur herstellen zu können. In einem ad hoc Netz kann jedoch nicht von einer Mindestanzahl ausgegangen werden.

4.3.3.2 Erzeugen von virtuellen Topologien

In diesem Abschnitt wird die Erzeugung von Baum- und Ringtopologie als virtuelle Topologie erläutert.

Baumtopologie: Zuerst wird die Überlagerung eines Segments mit einer Baumtopologie präsentiert. Ein Baum ist so strukturiert, dass n Knoten mit $n-1$ Kanten verbunden sind. Deshalb kann eine Information mit $n-1$ Nachrichten alle Knoten des Baumes erreichen.

Wenn ein Segment mit einem Baum überlagert werden soll, muss zuerst eruiert werden, ob dieser Vorgang für alle Segmente generell möglich ist. Als Beweis dient ein Satz aus der Graphentheorie [Die00].

Definition 6: *Spannbaum*
Ein Baum $T \subseteq G$ heißt Spannbaum von G, wenn er ganz G aufspannt, d. h. wenn $V(T) = V(G)$ ist.

□

Ein Spannbaum ist somit ein Gerüst eines zusammenhängenden Graphen, welches aus sämtlichen Knoten aber nicht allen Kanten besteht. Ein Spannbaum genügt damit der Definition 4.3.3.1 einer virtuellen Topologie. Dass jedes Segment mit einem Spannbaum überlagert werden kann, besagt folgender Satz:

Satz 1: *Existenz eines Spannbaumes*

Ein Graph G hat einen Spannbaum genau dann, wenn er zusammenhängend ist.

Beweis nach Tutte [Tut84]:
Hin-Richtung: Wenn G einen Spannbaum besitzt, so ist er zusammenhängend, da ein Baum per Definition immer zusammenhängend ist.
Rück-Richtung: G sei zusammenhängend und es werden sukzessive Kanten von G entfernt mit der Nebenbedingung, dass G auch weiterhin zusammenhängend ist. Wenn keine Kanten mehr entfernt werden können, ist der verbleibende Graph ein Baum.

■

Ein Segment ist ein zusammenhängender Graph, da in einem Segment nur Knoten enthalten sind, die miteinander kommunizieren können. Deshalb existiert für jedes Segment ein Spannbaum.

Im Weiteren wird die Erzeugung einer Baumstruktur erläutert. Für den Start des Algorithmus ist kein Initiator notwendig. Ist das Segment erzeugt, kann jeder Knoten

die Baumerzeugung anstoßen. Der Algorithmus kann auch von mehreren Knoten initiiert werden. Begonnen wird jedoch mit der Annahme, dass nur ein Knoten die Initiierung übernimmt.

Die Spannbaumerzeugung in verteilten Systemen ist bereits gut in der Literatur untersucht und gehört zu den verteilten Basisalgorithmen [Mat89].

Der präsentierte Algorithmus lehnt sich eng an den ECHO-Algorithmus [Cha82; Seg83] an. Es wird jedoch auf das explizite Senden der ECHO-Nachrichten verzichtet, da - sofern keine Rückantwort eintrifft - eine Kante implizit als Element des Baumes angesehen wird.

Für den Algorithmus verwaltet jeder Knoten eine Spannbaumteilnehmerliste (SBTL), in der er einträgt, welche Nachbarknoten der Baumstruktur angehören.

Der Initiator sendet an alle direkten Nachbarn eine `TREE_CREATE` Nachricht. Der Initiatorknoten wird hiermit der Wurzelknoten des Spannbaumes. Empfängt ein Knoten eine `TREE_CREATE` Nachricht und er ist noch nicht Mitglied der Baumstruktur, so wird der Sender als Elternknoten verzeichnet und reicht die `TREE_CREATE` Nachricht an seine weiteren Nachbarn weiter. Alle Knoten, an die er die Nachricht weiterleitet, werden in der SBTL vermerkt. Bekommt ein Knoten A, der bereits Teil der Baumstruktur ist, von einem Knoten B eine Nachricht, wird Knoten B aus der SBTL gelöscht. Auf diese Weise werden Zyklen aus dem Graphen genommen und ein Baum entsteht. Die Nachrichten werden nur innerhalb eines Segments weitergeleitet um eine Terminierung zu garantieren. Das Vorgehen entspricht der klassischen Tiefensuche in Graphen.

Es wurde bereits beschrieben, dass prinzipiell jeder Knoten des Segments die Erzeugung initiieren kann. Jedoch gibt es hierfür Knoten, die sich mehr für diese Aufgabe eignen als andere. Der Baum sollte eine minimale Tiefe (d. h. den Designparameter k bzw. $k+1$ aus Abschnitt 4.3.2.2) aufweisen. Hierdurch würden sich weniger Schritte bei der Erzeugung ergeben. Die Anzahl der versendeten Nachrichten ist hiervon jedoch nicht abhängig. Bei niedrigem k-Wert parallelisiert sich der Vorgang mehr, so dass er schneller abläuft.

Somit eignet sich der Initiator des Segments am besten für die Aufgabe der Initiierung der Baumstruktur, da hierdurch die Schritte bei der Erzeugung minimal sind. Aber auch die anderen inneren Segmentknoten kommen in Betracht. Zu vermeiden gilt es, dass Randknoten als Initiator fungieren, denn hierdurch würde sich die Erzeugungszeit verdoppeln.

Als Ergebnis kann somit festgehalten werden, dass jeder Nichtrandknoten - sofern er nicht Initiator des Segments war - auch die Baumstruktur initiieren kann. Da es durchaus auch Segmente geben kann, die nur aus Randknoten bestehen, kann nicht verallgemeinert werden, dass Randknoten nie Segmente initiieren.

Um einen Baum in einem Graphen $G = (V, E)$ zu erstellen sind $2|E| - |V| + 1$ Nachrichten nötig. Diese Zahl ergibt sich, da eine Nachricht über alle Kanten verschickt wird, wobei die eingehende Kante eine Ausnahme darstellt. Der Initiator versendet jedoch Nachrichten an alle ausgehenden Kanten, so dass dies durch die Addition mit 1 berücksichtigt werden muss. Mit $|E| = \frac{|V|(|V|-1)}{2}$ - dem Maximalwert von $|E|$ - ergibt sich eine Komplexitätsordnung von $\mathcal{O}(|V|^2)$.

In Abbildung 4.12 ist der Algorithmus im Pseudocode ersichtlich. Zu beachten ist, dass sich der Terminus *Nachbar* nur auf die Segmentmitglieder bezieht.

Abbildung 4.13 zeigt die Erzeugung der Baumstruktur an einem Beispielnetzwerk, welches in Abbildung 4.13a zu sehen ist. In Abbildung 4.13b initiiert Knoten 7 den Vorgang,

```
Initialisierung eines Knoten:
    SBTL = null;
    initiator = false;
    sent = false;
    root = null;

Start:
    initiator := true;
    sende TREE_CREATE Nachricht zu allen Nachbarn;

Bei Empfang einer TREE_CREATE Nachricht von Knoten p:
    if( not sent )
        root = p;
        SBTL = alle Nachbarn außer p;
        sende TREE_CREATE Nachricht zu allen Nachbarn außer p;
        sent = true;
    fi;
    else SBTL -= p;
```

Abbildung 4.12: Pseudocode für den Baumerzeugungsalgorithmus

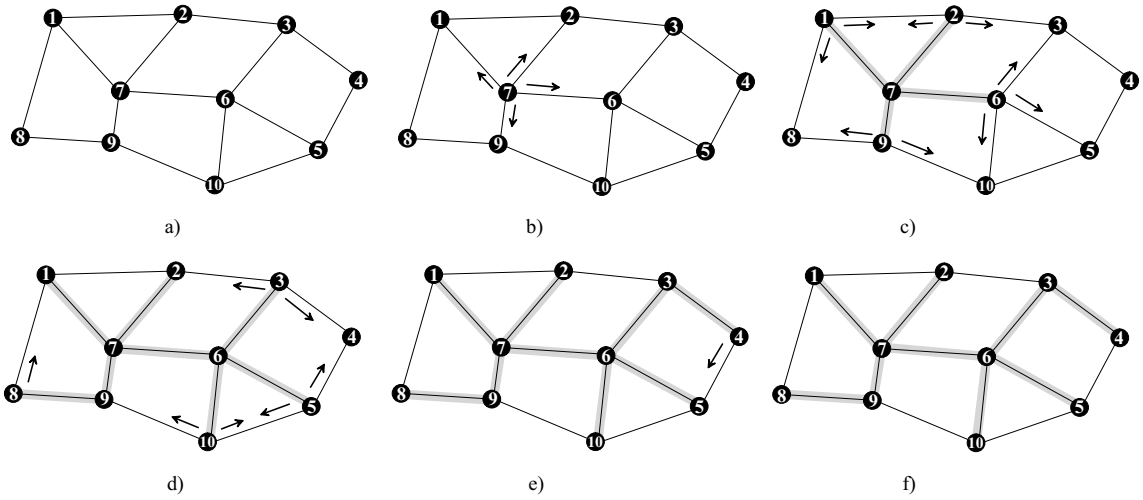


Abbildung 4.13: Sukzessive Erzeugung der Baumtopologie

indem er zu all seinen Nachbarn Nachrichten sendet. Da noch keiner der Nachbarn Mitglied der Baumstruktur ist, leiten diese die Nachricht weiter (Abbildung 4.13c). Ferner ist in Abbildung 4.13c zu sehen, dass Knoten 8 sowohl eine Nachricht von Knoten 1 als auch von Knoten 9 bekommt. Wesentlich ist in diesem Fall, welche Nachricht zuerst bei Knoten 8 eintrifft bzw. bearbeitet wird. Der entsprechende Senderknoten wird Elternknoten von Knoten 8. In Abbildung 4.13c wird angenommen, dass die Nachricht von Knoten 1

zuerst ankommt. Knoten 8 sendet daraufhin eine `TREE_CREATE` Nachricht an Knoten 1. Wenn diese Nachricht bei Knoten 1 eintrifft, wird Knoten 8 aus der SBTL von Knoten 1 gelöscht. Knoten 8 verfährt ebenso. Damit wird die Verbindung der Knoten 1-8 als nicht zur Baumstruktur gehörend gekennzeichnet.

Solche Fälle treten ebenso in Abbildung 4.13d noch auf, wobei in gleicher Weise verfahren wird. In Abbildung 4.13e ist die Bildung der virtuellen Baumtopologie beendet.

Der vorgestellte Algorithmus geht von einem Initiator aus. Bei mehreren Initiatoren wird das Nachrichtenauslöschungsprinzip (message extinction) angewandt [Mat89]. Dieses Verfahren besagt, dass von mehreren Initiierungen nur eine Nachricht weitergeleitet wird. Treffen bei einem Knoten Nachrichten von zwei Initiierungen ein, wird nur für eine der Algorithmus fortgeführt. Um zu entscheiden, welche Initiierung „überleben“ soll, wird ein Zähler mitgeführt, der ausgehend vom Initiator von Nachbar zu Nachbar dekrementiert wird. Bei mehreren Initiierungen wird sich ein Knoten für die Initiierung entschieden, deren Zähler niedriger ist, da die Topologie schon mehr Raum einnimmt als die andere. Im Falle eines gleichen Zählerstandes kann willkürlich entschieden werden.

Inwiefern bereits Teile anderer Initiierungen übernommen werden können, lässt sich nicht verallgemeinern und müsste deshalb vor der Neuentwicklung des Baumes überprüft werden. Diese Überprüfung nähme das Senden ebenso vieler Nachrichten in Anspruch, wie die vollständige Neuentwicklung einer Baumstruktur, weshalb in der vorliegenden Implementierung die Ergebnisse der anderen Initiierungen verworfen werden.

Ringtopologie: Eine Alternative zur Baumtopologie ist eine Überlagerung eines Segments mit einer Ringstruktur. Mit einem Ring kann wie bei der Baumstruktur mit $n-1$ Nachrichten eine Information an alle Teilnehmer gesendet werden. Bezüglich der Anwendbarkeit sei vorweggenommen, dass der Aufbau einer virtuellen Ringstruktur nur lohnend ist, wenn die Konnektivität des Netzwerks groß ist.

Bei der Ringstruktur muss zwischen einem physikalischen und einem logischen Ring unterschieden werden. In einem logischen Ring sind die Nachfolger und Vorgänger eines Knoten nicht notwendigerweise seine direkten Nachbarn, wogegen im physikalischen Ring die Vor- und Nachfolger jeweils direkte Nachbarn eines Knotens sind. Graphentheoretisch entspricht ein physikalischer Ring einem Hamiltonkreis.

Auch hinsichtlich der Existenz gibt es Unterschiede. Ein Netzwerk muss nicht mit einem physikalischen Ring überlagert werden können. Die Graphentheorie kennt eine Fülle von Sätzen, die aussagen, welche Bedingungen erfüllt sein müssen, damit ein Graph hamiltonsch ist. (siehe hierzu z. B. Kapitel 8 in Diestel [Die00]). Als Gegenbeispiel kann beispielsweise ein Baum angesehen werden, der nicht hamiltonsch ist. Ein logischer Ring lässt sich jedoch immer realisieren, da bezüglich des Nachfolgeknotens keine Einschränkungen bestehen.

Ziel soll es jedoch sein, einen Ring mit möglichst wenig logischen Kanten zu konstruieren, da bei logischen Ringen die Nachrichtenanzahl von $n-1$ nicht erreicht wird. Zu bedenken ist auch, dass der Algorithmus seitens seiner Komplexität beherrschbar sein muss. Die Erzeugung eines Hamiltonkreises entspricht dem *Travelling Salesman* Problem und ist somit $\mathcal{NP}$-vollständig [Bra94]. Es gibt jedoch eine Möglichkeit unter gewissen Voraussetzungen diese Komplexität zu verringern, welche nachfolgend erläutert wird.

Grundprinzip des Ringerzeugungs-Algorithmus ist es, dass jeder Knoten einen Ring

erzeugt, der aus möglichst vielen Nachbarn und deren Nachbarn besteht. Dadurch dass jeder Knoten einen Ring erzeugt, soll eine globale Struktur erreicht werden, die die Anzahl an Verbindungsknoten reduziert und einem globalen Ring möglichst nahe kommt.

Charakteristisch für einen Ring ist, dass ein Knoten genau zwei Nachbarn und somit genau zwei ausgehende Kanten besitzt. Die Anzahl inzidenter Kanten eines Knoten n wird als Grad von n bezeichnet. Hat ein Segmentknoten genau den Grad zwei, ist genau festgelegt, welche Kanten zur Ringerzeugung mit einbezogen werden. Knoten mit nur einer Verbindung - wie z. B. Knoten 7 in Abbildung 4.13 - nehmen an der lokalen Ringerzeugung nicht teil, da hierdurch kein Mehrwert entsteht.

Handlungsbedarf besteht deshalb primär, wenn ein Knoten einen Grad größer als zwei besitzt. In diesem Fall sollen die vorhandenen Kanten in der virtuellen Topologie reduziert werden, um weniger Nachrichten für eine Einigung senden zu müssen.

Um den lokalen Ring erzeugen zu können, sendet jeder Knoten seine direkten Kommunikationspartner an all seine Nachbarn[6]. Jeder Knoten verwaltet diese Daten in einer Adjazenzmatrix, in der angegeben ist, welche Knoten miteinander verbunden sind. Nachdem die notwendigen Daten vorhanden sind, muss jeder Knoten einen Ring (Kreis) mit möglichst vielen Knoten erzeugen. Ob und unter welchen Umständen dies möglich ist, dazu gibt folgender Satz Auskunft.

Satz 2: *Existenz eines Ring (Kreises)*
Jeder Graph G mit e Kanten und n Knoten vom Minimalgrad $\delta(G) \geq 2$ enthält einen Kreis der Länge mindestens $\delta(G) + 1$. Ein Graph von der Dichte $d := \frac{e}{n} \geq 2$ enthält einen Kreis der Länge $> d$. (Beweis siehe Brandstädt [Bra94].)

Um aus diesem Satz den notwendigen Algorithmus herzuleiten, ist der Begriff und die Notation der Nachbarschaftsmenge notwendig.

Definition 7: *Kommunikationsrelation $v_i \, \mathcal{C}_n \, v_j$*
Die Relation $v_i \, \mathcal{C}_n \, v_j$ gibt an, dass ein Knoten v_i über maximal n Kanten mit Knoten v_j kommunizieren kann, d. h. es gibt einen Pfad $\{e_1, \ldots, e_m\}$ $m \leq n$ von v_i nach v_j in G mit maximaler Länge n.

□

Definition 8: *Nachbarschaftsmenge N_v^k*
$N_v^k = \{x \mid v \, \mathcal{C}_k \, x\}$. Somit gibt N_v^k die Menge der Knoten an, mit denen der Knoten v über k Kanten kommunizieren kann.

□

Ein Graph $G' = (V', E')$ heißt Teilgraph von $G = (V, E)$, wenn $V' \subseteq V$ und $E' \subseteq E$ gilt. Ein Teilgraph heißt *induziert*, wenn er alle Kanten $(v_i, v_j) \in E$ mit $v_i, v_j \in V'$ enthält. Dieser induzierte Teilgraph sei mit $G[V']$ bezeichnet.

Dann sei $G' = G[N_v^2]$ der durch N_v^2 induzierte Graph. $G' = (V', E')$ ist somit der Graph, der sich durch die Nachbarn eines Knoten v und deren Nachbarn ergibt. Um einen Kreis in $G' = (V', E')$ zu erzeugen, wird ausgehend von einer Kante $\{v_0, v_1\} \in E'$, dieser Pfad $P = \{v_0, v_1\}$ so oft verlängert, bis sämtliche Nachbarn des Endpunktes v_l von $P = \{v_0, v_1, \ldots, v_l\}$ selbst in P liegen. Der so erhaltene Kreis C muss den Knoten v selbst nicht enthalten ($C \subseteq P \subseteq V'$).

[6]Die Nachbarn sind die direkten Kommunikationspartner

Ein Kreis könnte alternativ auch mit Hilfe der Tiefensuche auf Graphen [Mei91] entwickelt werden. Die Tiefensuche ist eine Verallgemeinerung der Traversierung von binären Bäumen in Pre-Order.

All diese erwähnten Verfahren garantieren jedoch nicht, dass der Kreis (Ring) möglichst groß ist. Soll ein Kreis mit möglichst vielen Knoten entstehen, nähert sich dieses Problem dem des Findens eines Hamiltonkreises in einem Graphen. Ein Hamiltonkreis ist ein Kreis der alle Knoten aus V enthält. Um generell zu eruieren, ob eine Knotenmenge V durch einen Kreis verbunden werden kann, hilft folgender Satz:

Satz 3: *(Chvátal 1972)*
Sei $G = (V, E)$ ein endlicher Graph mit $n = |V| \geq 3$ Knoten. Die Knoten $v_1, \ldots, v_n$ seien so nummeriert, dass ihre Grade (d. h. die Anzahl an ausgehenden Kanten) eine aufsteigende Kette $d_1 \leq \ldots \leq d_n$ bilden. Wenn für alle natürlichen Zahlen k, $1 \leq k \leq \frac{n}{2}$ die Implikation

$$d_k \leq k \Rightarrow d_{n-k} \geq n - k \tag{4.2}$$

erfüllt ist, dann besitzt G einen Hamiltonkreis [Bra94].

Zuerst kann für die komplette Nachbarschaftsmenge N_v^2 eines Knotens v mit Hilfe von Satz 4.3.3.2 errechnet werden, ob es einen Hamiltonkreis gibt. Ist dies nicht der Fall, so wird sukzessive jeweils der Knoten mit dem geringsten Grad entfernt.

In diesem Satz wird jedoch keine Aussage gemacht, wie der Hamiltonkreis gefunden werden kann. Für eine kleine Menge an Knoten kann nach dem Zufallsprinzip vorgegangen werden. Da im schlechtesten Fall jedoch $(n-1)!$ Möglichkeiten begutachtet werden müssen, ist schon bei einer sehr geringen Anzahl an Knoten diese Methode ausgereizt.

Um ein effizientes Verfahren vorzustellen, welches einen Hamiltonkreis erzeugt, sind jedoch noch grundlegende Begriffe der Graphentheorie notwendig.

Definition 9: *Zusammenhangszahl $\kappa(G)$ eines Graphen*
Sei $G = (V, E)$ ein Graph, dann ist $\kappa(G) = \min\{|T| : T \subset V$ und $G \backslash T$ ist nicht zusammenhängend $\}$.

Ein Graph heißt k-fach-zusammenhängend, wenn $\kappa(G) \geq k$ gilt [Jun94].

□

Definition 10: *Unabhängigkeitsbegriff in der Graphentheorie*
Eine Teilmenge X der Knotenmenge V eines Graphen $G = (V, E)$ heißt *unabhängig*, wenn keine zwei Punkte von X durch eine Kante verbunden sind.

Die maximale Mächtigkeit $\alpha(G)$ einer unabhängigen Knotenmenge von G wird als *Unabhängigkeitszahl* von G bezeichnet [Jun94].

□

Mit diesen Definitionen kann folgender Satz erklärt werden:

Satz 4: *Chvátal, Erdős, 1972*

Für einen Graphen G der Ordnung ≥ 3 gilt:

$$\kappa(G) \geq \alpha(G) \Rightarrow G \text{ ist hamiltonsch.}$$

Ein Beweis ist in Brandstädt oder Jungnickel [EWHK$^+$96; Jun94] zu finden.

Obiger Satz wird aus diesem Grund erwähnt und ist bemerkenswert, da - obwohl das Auffinden eines Hamiltonkreises i. A. $\mathcal{NP}$-vollständig ist - bei Vorliegen der im Satz erwähnten Bedingung $\kappa(G) \geq \alpha(G)$ ein polynomieller Algorithmus ($\mathcal{O}(n \cdot e)$) existiert. Die Bedingung $\kappa(G) \geq \alpha(G)$ kann fast (vorausgesetzt $n \geq 3$) immer erfüllt werden, indem nur Knoten mit hohem Grad zur Ringbildung herangezogen werden, so dass α möglichst gering ist.

Der Kreis C, der aus obigem Verfahren resultiert, muss noch erweitert werden, um möglichst viele Knoten in G' zu verbinden. Dies geschieht in Anlehnung des konstruktiven Beweises von Satz 4.3.3.2 [EWHK$^+$96].

S sei durch $G[V' \backslash C]$ gegeben und T eine Knotenmenge, definiert durch $T = C \cap \bigcup_{s \in S} N_s^1$. T stellt die Menge der Knoten auf C dar, die Nachbarn in S haben. C sei gegeben durch $C = (v_1, \ldots, v_l)$. Aus T sind gemäß der Vorgaben zwei Knoten v_a und v_b durch einen Pfad miteinander verbunden, der nur innere Knoten aus S benutzt. Ist $v_a \in C$ und entspricht der Nachfolger von $v_a \in C$ dem v_b, so kann (v_a, v_b) aus C durch den Pfad aus S erweitert werden. Da $n \geq 3$ muss dieses Verfahren maximal (n-3) mal wiederholt werden um einen vollständigen Hamiltonkreis zu erhalten.

In Abbildung 4.14 ist ein Beispiel gegeben. Abbildung 4.14a zeigt einen Beispielgraph

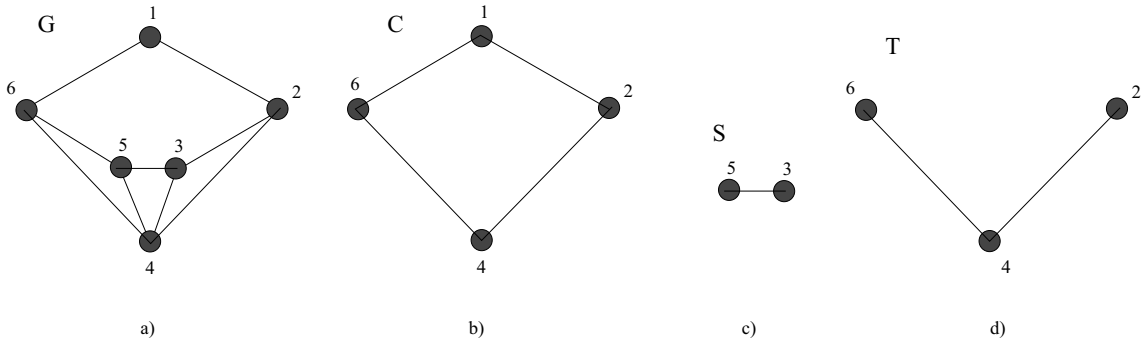


Abbildung 4.14: Algorithmus gemäß Satz von Chvátal und Erdős

$G_B = (V_B, E_B)$ in dem ein Hamiltonkreis gefunden werden soll. G_B ist 2-zusammenhängend, da es von jedem Knoten zu jedem anderen Knoten zwei kreuzungsfreie Wege gibt, so dass $\kappa(G_B) = 2$. Ferner sind nie mehr als zwei Knoten unabhängig, so dass $\alpha(G) = 2$ gilt und somit die Anforderungen an den Satz von Chvátal und Erdős erfüllt sind.

In Abbildung 4.14b ist ein Kreis C wiedergegeben, der einen Teilgraph von G_B darstellt und der Ausgangspunkt für den Hamiltonkreis ist. Abbildung 4.14c spiegelt die Knotenmenge $G[V_B \setminus C]$ wider und Abbildung 4.14d die Menge T, die sämtliche Knoten von C enthält, die Nachbarn in S haben.

Mit Hilfe dieser Knotenmenge wird versucht, den anfänglichen Kreis sukzessive zu einem Hamiltonkreis zu vergrößern. In Abbildung 4.15 ist ein Schritt des Verfahrens gezeigt. Existieren in T zwei benachbarte Knoten v_i und v_i^* und sind auch diese in C benachbart, so kann durch einen Knoten aus S der Kreis C vergrößert werden. Dieser Vorgang muss so oft wiederholt werden, bis S leer ist. Ist die Bedingung $\kappa(G) \geq \alpha(G)$ erfüllt, so wird als Resultat ein Hamiltonkreis erhalten.

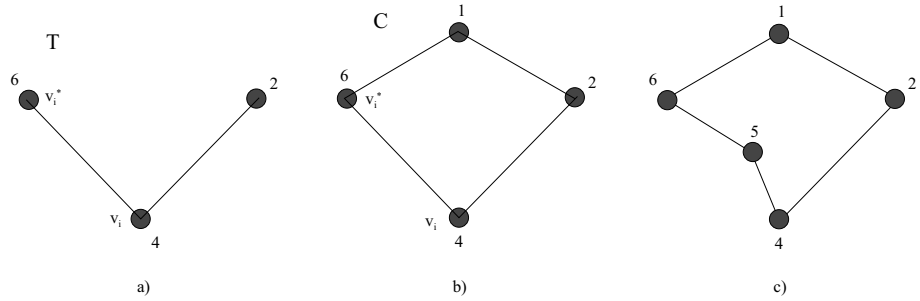


Abbildung 4.15: Vergrößern des Kreises

War die Erstellung eines Rings möglich, wird dies mit der ersten Nachricht allen Nachbarn mitgeteilt. Von diesem Zeitpunkt an werden Nachrichten nur noch entlang der Ringkanten versendet. Kommt kein Ring zustande, kann auch die Knotenmenge nicht reduziert werden. Dies tritt jedoch nur auf, wenn die Konnektivität gering und eine virtuelle Topologie nicht notwendig ist.

Zu sehen ist die Erstellung des Pseudorings in Abbildung 4.16a - 4.16l. In Abbildung 4.16b ist der entstandene Kreis zu sehen. Zu der N_v^2 zählen die Knoten 2, 3, 5, 8, 9, die bis auf Knoten 9 alle im Kreis vorhanden sind. In den folgenden Bildern ist für jeden Knoten diese Prozedur zu sehen. Für Knoten 9 kann kein Ring gebildet werden. Für die Knoten 7 und 10 muss es nicht versucht werden, da diese nur eine Kante besitzen.

Werden diese Informationen schließlich ausgetauscht und miteinander verglichen, können einige Kanten des Graphen eliminiert werden und es entsteht die virtuelle Topologie, wie in Abbildung 4.16l ersichtlich.

4.3.3.3 Aufrechterhalten der Topologie

Auf Grund der Mobilität der Knoten reicht die Erstellung einer virtuellen Topologie nicht aus, da sich Knoten entfernen und neue hinzukommen können. Aus diesem Grund werden für obige Topologien Mechanismen vorgestellt, wie diese in dynamischer Umgebung aufrecht erhalten werden können, und es wird dargestellt, wie groß der Aufwand hierfür ist.

Bei der Aufrechterhaltung muss zwischen zwei Fällen unterschieden werden.

1. Es kommen neue Knoten hinzu oder existierende Knoten verschwinden aus dem Segment. In diesem Fall ändert sich die Anzahl der Knoten im Segment.

2. Knoten müssen innerhalb des Segments umarrangiert werden, d. h. die Baum- oder Ringstruktur ändert sich, weil existierende Verbindungen innerhalb der Topologie abbrechen. In diesem Fall muss sich die Knotenzahl im Segment nicht ändern. Sie kann sich jedoch ändern, wenn durch die Änderungen andere Segmentknoten vom Segment abgetrennt werden.

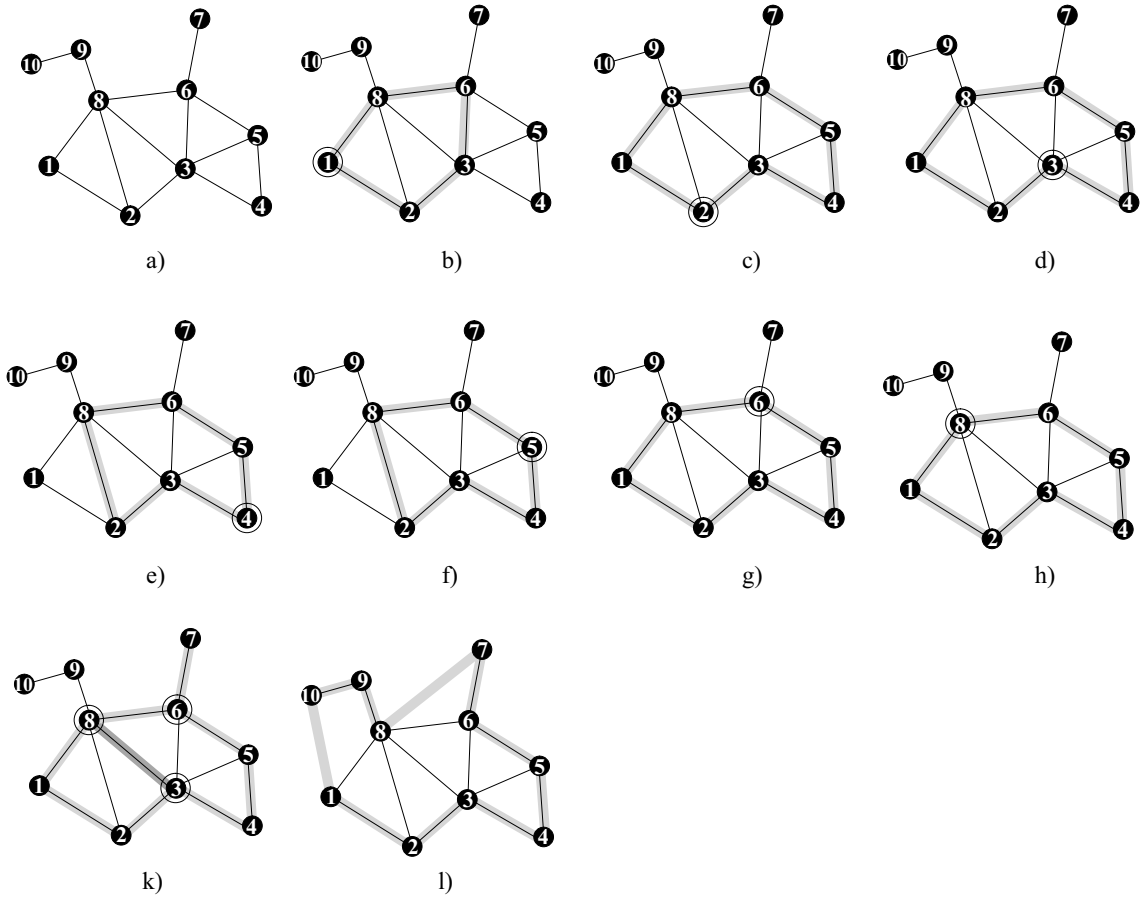


Abbildung 4.16: Erzeugung eines virtuellen Ringes in einem Netzwerk

Baumtopologie: Der einfachste Fall ist das Hinzufügen von Knoten. Hier ist besonders darauf zu achten, dass keine Ringe entstehen, die so die Baumstruktur zerstören. Aus diesem Grund darf ein neuer Knoten nur zu einem existierenden Baumknoten eine neue virtuelle Verbindung aufnehmen. Wurde ein Knoten neu in das Segment aufgenommen, so sendet dieser an alle Nachbarn, die Segmentteilnehmer sind, eine `TREE_MEMBER_ADD_REQUEST` Nachricht. Ein Empfänger dieser Nachricht sendet eine `TREE_MEMBER_ADD_RESPONSE` Nachricht zurück, wobei sein k Parameter zusätzlich mit der Nachricht übertragen wird. Der neu hinzugekommene Knoten wählt den Anschlussknoten aus, der den niedrigsten k Wert besitzt, um so möglichst zentral in der Struktur positioniert zu sein. Diesem Knoten wird eine `TREE_MEMBER_ADD_ACCEPT` - den nicht gewählten eine `TREE_MEMBER_ADD_REJECT` Nachricht gesendet.

Es bedarf folglich dreier Schritte, um einen neuen Knoten zur Topologie hinzuzufügen. Sollte der neue Knoten mehreren Segmenten angehören, wird dieser Vorgang unabhängig in allen Segmenten durchgeführt. In Abbildung 4.17 ist der Pseudocode dieses Verfahrens ersichtlich.

```
Neuer Knoten:

        Init k_max := -1;
        receiver = null;
        send TREE_MEMEBER_ADD_REQUEST to all neighbors;
        wait for TREE_MEMBER_ADD_RESPONSE messages;

    Bei Empfang von TREE_MEMBER_ADD_RESPONSE:
        read Parameter k;
        if k > k_max
            k_max = k;
            receiver = getReceiver();
        fi;

    Nach Empfang aller TREE_MEMBER_ADD_RESPONSE messages:
        send TREE_MEMBER_ADD_ACCEPT to receiver;
        send TREE_MEMBER_ADD_REJECT to all others;

Segmentmitglied:

    Bei Empfang von TREE_MEMEBER_ADD_REQUEST:
        send TREE_MEMBER_ADD_RESPONSE
            with Parameter k;
```

Abbildung 4.17: PseudoCode für die Aufrechterhaltung der Baumtopologie

Entfernt sich ein Knoten, sind die durchzuführenden Aktionen abhängig von der Position des Knoten im Baum. Entfernt sich ein Blattknoten von der Topologie, sind keine expliziten Aktionen notwendig. Der Knoten, der mit dem Blattknoten verbunden war, eliminiert diesen aus seiner Spannbaum-Teilnehmerliste. Dies ist gleichzeitig der häufigste Fall, da sich i. A. die Randknoten aus dem Segment entfernen. Der ehemalige Blattknoten muss sich, wie bereits oben beschrieben, um die Aufnahme in ein neues Segment und in eine neue Baumtopologie bemühen.

Doch es kann auch geschehen, dass Nichtrandknoten aus dem Segment „ausreißen“. So kann es sich geschehen, dass ein Teilnehmer sein mobiles Gerät ausschaltet und der Knoten somit nicht mehr im Segment existiert.

Nachdem ein Knoten aus dem Netz verschwunden ist, sendet jeder hiervon betroffene Knoten eine `TREE_ADD_NODE_REQUEST` Nachricht. Die Nachricht sendet ein Knoten, wenn bestehende Baumtopologiekanten entfernt werden und der Knoten noch weitere zum Segment gehörende Kanten aufweist. Entlang dieser Kanten wird die `TREE_ADD_NODE_REQUEST` Nachricht gesendet. Empfängt ein Knoten solch eine Mitteilung, so sendet dieser eine `TREE_ADD_NODE_PROPOSAL` Nachricht zurück, wenn er Teil der Baumtopologie ist. Ist dies nicht der Fall, leitet er die Nachricht an seine Nachbarn weiter, sofern welche vorhanden sind und auch noch nicht gesendet wurden. Empfängt ein Knoten `TREE_ADD_NODE_PROPOSAL` Nachrichten, muss er genau einen Senderknoten auswählen und zu diesem eine `TREE_ADD_NODE_ACCEPT_PROPOSAL` Nachricht senden. Hiermit wird eine neue Kante der

Baumtopologie erzeugt und der Empfängerknoten wird in die SBTL eingetragen. Diesen Eintrag muss auch der Empfänger der Nachricht für den Sender vornehmen. Eine TREE_ADD_NODE_PROPOSAL Nachricht muss weitergeleitet werden, wenn der empfangende Knoten nicht Blattknoten war und weitere Segmentkanten existieren. Diese Weiterleitung bricht ab, wenn ein Knoten bereits wieder Mitglied des Baumes ist oder sie bereits weitergeleitet hat.

In jeder Nachricht muss der Identifikator des entschwundenen Knotens enthalten sein. Hintergrund ist, dass sich mehrere Knoten quasi gleichzeitig entfernen können und deshalb mehrere TREE_ADD_NODE_*XXX* Nachrichten im Umlauf sein können. Der Pseudocode ist in Abbildung 4.18 widergegeben.

```
init:
    treeAddNodeProposalSent=false;

Betroffener Knoten:
        send TREE_ADD_NODE_REQUEST;

    Nach Empfang aller TREE_ADD_NODE_PROPOSAL messages
    bzw. nach Ablauf des Timeouts:
        receiver = selectReceiver();
        send TREE_ADD_NODE_ACCEPT_PROPOSAL to receiver;
        add receiver to SBTL;

Nichtbetroffene Segmentknoten:

    Bei Empfang von TREE_ADD_NODE_REQUEST:
        if node is element of tree topology
            send TREE_ADD_NODE_PROPOSAL to sender;
        else
            send TREE_ADD_NODE_REQUEST to all neighbors;

    Bei Empfang einer TREE_ADD_NODE_PROPOSAL message:
        if( !treeAddNodeProposalSent && node is leafnode )
            send TREE_ADD_NODE_PROPOSAL to segmentmembers;
            treeAddNodeProposalSent=true;
        fi;

    Bei Empfang von TREE_ADD_NODE_ACCEPT_PROPOSAL
        add sender to SBTL;
```

Abbildung 4.18: PseudoCode für die Entfernung eines Nichtrandknotens

Das Beispiel in Abbildung 4.19 zeigt den Ablauf, wenn Knoten 7 aus Abbildung 4.19a entfernt wird. In Abbildung 4.19c senden die Knoten 1, 2 und 9 TREE_ADD_NODE_REQUEST Nachrichten entlang Segmentkanten, die nicht gleichzeitig in der SBTL enthalten sind. Die Knoten, die der Baumstruktur angehören, senden in Abbildung 4.19d TREE_ADD_NODE_PROPOSAL Nachrichten zurück. Knoten 9 bestätigt in Abbildung 4.19e mit einer TREE_ADD_-

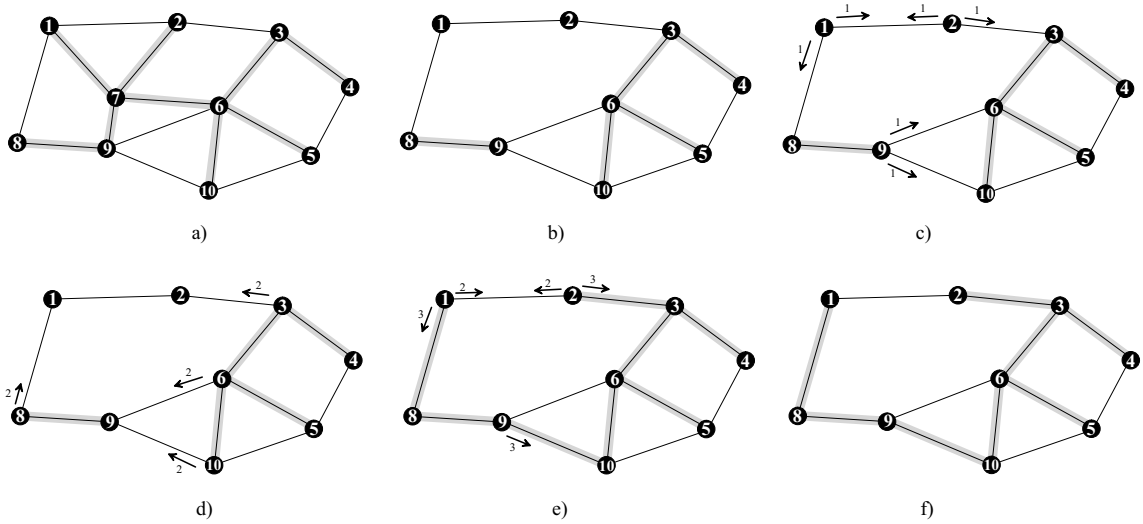


Abbildung 4.19: Beispiel für das Verschwinden eines Nichtrandknotens. Zur Unterscheidung der Nachrichten wurden diese mit Ziffern über den Pfeilen versehen. Mit der Nummer 1 sind `TREE_ADD_NODE_REQUEST`, mit der Ziffer 2 `TREE_ADD_NODE_PROPOSAL` und mit 3 `TREE_ADD_NODE_ACCEPT_PROPOSAL` Nachrichten gekennzeichnet.

`NODE_ACCEPT_PROPOSAL` Nachricht an Knoten 10 das Erzeugen einer neuen Baumkante. Knoten 1 und 2 verfahren ähnlich. Außerdem leiten die Knoten die `TREE_ADD_NODE_ACCEPT_PROPOSAL` Nachricht weiter. Da diese Nachricht von beiden Knoten auch empfangen wird, ist die Verbindungskante keine Kante der virtuellen Topologie. Abbildung 4.19f zeigt die aktualisierte Baumstruktur.

Entfernt sich ein Knoten gänzlich aus dem Segment, kann es geschehen, dass sich die virtuelle Topologie in zwei Teile aufteilt, wenn es sich bei dem betreffenden Knoten um einen Verbindungsknoten handelte. Dies ist anhand eines konstruierten Beispiels in Abbildung 4.20 zu sehen. Dadurch dass Knoten 6 aus dem Netz verschwindet, teilt sich die

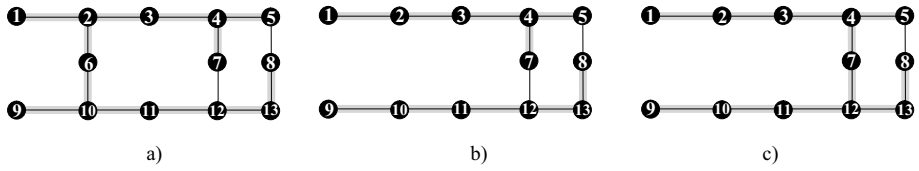


Abbildung 4.20: Beispiel für das Verschwinden eines Nichtrandknotens mit Teilung der Baumtopologie

Baumtopologie in zwei Teile auf. Wird von einem oder mehreren Knoten entdeckt, dass zu einem Knoten kein Kontakt mehr besteht, muss somit trotz der Tatsache, dass diese

Knoten noch Element der Baumtopologie sind, überprüft werden, ob der Baum noch als zusammenhängende Struktur vorliegt. Um dies zu kontrollieren senden die entdeckenden Knoten eine `TREE_CONNECTION_CHECK` Nachricht an alle ihre Nachbarn - unabhängig ob diese bereits Topologieteilnehmer sind. In der Nachricht muss der Identifikator des Entdeckerknotens enthalten sein. Wenn es mindestens zwei Initiatoren gibt und die Topologie getrennt ist, dann muss zu einem Zeitpunkt ein Knoten eine Nachricht von beiden Initiatoren (hierfür sind die mitgesendeten Identifikatoren notwendig) erhalten, da ein Segment stets zusammenhängend ist. Von solch einem Knoten - im weiteren Vereinigungsknoten genannt - kann die Lücke geschlossen und Topologie verbunden werden. Doch es können mehrere potenzielle Verbindungsstücke existieren, wie ebenfalls in Abbildung 4.20 zu sehen ist. Folglich dürfen nicht alle Verbindungsstücke vereinigt werden, da ansonsten Zyklen in der Topologie auftreten und damit kein Baum entsteht.

Ein Vereinigungsknoten sendet über die Eingangskante der ersten `TREE_CONNECTION_CHECK` Nachricht und über sämtlichen Baumkanten eine `TREE_CYCLE_CHECK` Nachricht. Der Nachricht werden die Identifikatoren des Verbindungsknotens und der Entdeckerknoten beigefügt. Nimmt ein Knoten diese Mitteilung entgegen und der Sender ist nicht Teil der SBTL, so wird er eingetragen. Erhält ein Knoten eine `TREE_CYCLE_CHECK` Nachricht, leitet er sie an alle Knoten weiter, die Element der SBTL sind. Wird diese Nachricht ein weiteres Mal mit niedrigerem Wert für Identifikator des Verbindungsknoten (aber gleichen Werten für die Identifikatoren der Entdeckerknoten) empfangen, bedeutet dies, dass die Eintragungen in die SBTL wieder zurückgenommen werden müssen um Zyklen zu vermeiden. Möglich wird das durch eine zusätzlich `UNDO`-Tabelle, in die ein Knoten neben der SBTL eingetragen wird. Auf diese Weise können getrennte Topologien wieder vereint und Zyklen vermieden werden.

Dieser beschriebene Vorgang muss somit stets durchgeführt werden, wenn sich ein Knoten aus dem Netz entfernt. In Abbildung 4.19 müsste Knoten 6 den Vorgang initiieren, da zwar Knoten 6 noch mit dem Baum verbunden ist, aber nicht bekannt ist, ob durch den Verlust von Knoten 7 der Baum noch zusammenhängend ist. Abbildung 4.21 zeigt den Pseudocode des Verfahrens.

Im Beispiel aus Abbildung 4.20 bemerken Knoten 10 und Knoten 2, dass Knoten 6 sich aus dem Netz entfernt hat. Folglich senden beide eine `TREE_CONNECTION_CHECK` Nachricht an alle Nachbarn. In den Knoten 7 und 12 begegnen sich die `TREE_CONNECTION_CHECK` Nachrichten, was auf zwei getrennte Bäume hindeutet. Doch auch in den Knoten 5 und 8 werden sich die Mitteilungen begegnen, womit auch diese Kante als potenzielle Verbindung in Betracht kommt. Wenn jedoch `TREE_CYCLE_CHECK` Nachrichten gesendet werden, wird nur die Nachricht von Knoten 12 nicht gelöscht, womit sich die gegebene Lösung in Abbildung 4.20c ergibt.

Hinsichtlich der Komplexität beginnen zwei Knoten eine Nachricht zu senden, die von ihren Empfängern weitergeleitet wird. Da es einen Knoten gibt, in dem sich die Nachrichten treffen, traversiert jede der zwei Nachrichten die Hälfte der Anzahl der Knoten, so dass die Komplexitätsordnung linear ($\mathcal{O}(|V|)$ in der Anzahl der Knoten ist.

Eine komplette Neuinitiierung des Baumes wäre auch möglich, müsste jedoch von allen Knoten gestartet werden, die den Verlust entdecken. Dies würde eine noch aufwändigere Operation (Ordnung $\mathcal{O}(|V|^2)$ darstellen.

Wie bereits erwähnt, kann eine Änderung auch innerhalb eines Segments auftreten.

```
Initialisierung jedes Knotens:
    treeConnMsgReceived = 0;
    IDMin = MAX_INTEGER; // i.a. 2^31-1;

Knoten die den Verlust eines anderen Knoten bemerken:
    send TREE_CONNECTION_CHECK to neighbors

Bei Empfang von TREE_CYCLE_CHECK:
    id=getID();
    if (id <IDMin && sender is element of UNDO Table)
        remove sender from SBTL;
        remove sender from UNDO Table;
    fi;

    if Sender not in SBTL
        Enter sender to SBTL;
        Enter sender to UNDO Table;
    fi;

Bei Empfang von TREE_CONNECTION_CHECK:
    treeConnMsgReceived++;
    if( treeConnMsgReceived == 2)
        send TREE_CYCLE_CHECK;
    fi;
```

Abbildung 4.21: PseudoCode für die potenzielle Aufteilung einer Baumtopolgie

Da jeder Knoten die Teilmenge der Nachbarn kennt, die Element der Baumstruktur sind, entsteht somit Handlungsbedarf, wenn sich an dieser Menge etwas ändert. Dieser Fall kann jedoch als Kombination von Entfernung des betroffenen Knoten mit anschließendem neuem Hinzufügen angesehen werden, was bereits oben beschrieben wurde.

Als Letztes soll noch der Fall betrachtet werden, wenn sich zwei Segmente treffen. Da die Segmente verschmolzen werden, muss dies auch mit den Topologien geschehen. Beide Bäume könnten miteinander verbunden werden, wenn sichergestellt werden kann, dass die Verbindung mit nur einer Kante erfolgt. Bereits bei einer Verbindung mit zwei Kanten entstehen unerwünschte Zyklen. Die Überprüfung, dass nur eine Kante die Bäume verbindet, benötigt hohen Aufwand.

In der vorliegenden Implementierung wird an dieser Stelle eine neue Baumerzeugung für ein Segment initiiert, d. h. ein Baum wird erhalten und erweitert, der andere Baum verworfen. Die Problematik besteht in der Wahl, welcher Baum erhalten wird. Die Größe (k-Parameter) kann nicht in Betracht gezogen werden, da verschiedene Blattknoten der Bäume auch verschiedene Werte hierfür aufweisen können. Somit würde bei deren Verwendung keine Eindeutigkeit vorliegen. Bleibt als Entscheidungskriterium nur noch der Identifikator des Initiators der Baumerzeugung. Dieser ist eindeutig, muss aber nicht optimal sein. Da jedoch ein Baum erhalten bleibt, ist dieses Vorgehen dem der kompletten Neuinitiierung immer vorzuziehen. Die Initiierung erfolgt ausgehend von den Verbindungs-

knoten der beiden Segmente, wobei wie in Abschnitt 4.3.3.2 vorzugehen ist.

Abschießend kann zusammengefasst werden, dass es zwar wenig Aufwand bedeutet einen neuen Knoten in die Baumtopologie aufzunehmen. Dieser Aufwand ist von fester Komplexität $\mathcal{O}(1)$, da der Vorgang immer zwei Schritte umfasst.

Die Komplexität nimmt zu, wenn Knoten verschwinden und es sich nicht um Blattknoten handelt. Vor allem die notwendige Überprüfung auf den Zusammenhang der Struktur, nachdem sich ein Knoten entfernt hat, ist von linearer Ordnung $\mathcal{O}(|V|)$.

Ringtopologie: Auch bei der Ringtopologie müssen Mechanismen für die Aufrechterhaltung der Topologie bereitgestellt werden.

Im einfachsten Fall entfernt sich ein Knoten mit nur einer realen Verbindung, wie z. B. Knoten 7 in Abbildung 4.16. Beim Eintreten eines solchen Ereignisses ist hinsichtlich der Ringknoten keine Aktion notwendig. Sollte sich solch ein Knoten dem Segment anschließen, muss nur der betreffende Ringknoten den neuen Knoten als zusätzlichen Ringknoten aufnehmen.

Auch wenn ein Knoten mit weiteren Verbindungen, wie z. B. Knoten 9 in Abbildung 4.16, die Ringstruktur verlässt, besteht kein Handlungsbedarf. Für den betroffenen Ringknoten (in diesem Fall Knoten 8) ist nicht ersichtlich, ob durch das Verlassen von Knoten 9 noch weitere Knoten abhanden kommen.

Maßnahmen müssen eingeleitet werden, wenn sich ein Ringknoten aus dem Segment entfernt. Bemerkt ein Ringknoten, dass sich einer seiner Nachbarn - die gleichzeitig Teil der Ringstruktur sind[7] - entfernt, so ist der Ring unterbrochen und es muss nach einem Knoten gesucht werden, der diese Lücke schließt.

In diesem Fall muss erneut für diesen Knoten eine Ringstruktur erzeugt werden. In einem ersten Schritt müssen zur Bestimmung der Nachbarschaftsmenge N_v^2 die notwendigen Knoteninformationen beschaffen werden. Auf ein Anfordern der Nachbarknoten der direkten Kommunikationspartner kann nicht verzichtet werden, da in den noch vorliegenden Daten, die zur vorherigen Ringerzeugung nötig waren, Knoten eingetragen sein können, die sich bereits aus dem Segment entfernt haben. Nachdem diese Ergebnisse vorliegen, wird wie in Abschnitt 4.3.3.2 verfahren.

Auch eine Neuaufnahme eines Knotens in die Ringstruktur ist leicht zu bewerkstelligen, wenn die notwendigen Bedingungen hierfür erfüllt sind. Ein neuer Knoten muss eine Verbindung zu zwei[8] benachbarten Ringknoten herstellen, um sich anschließend zwischen diesen einfügen zu können. Somit sendet ein „neuer Knoten“ an all seine direkten Knoten eine `RING_MEMBER_ADD_REQUEST` Nachricht. Ist ein Knoten nicht Teil einer virtuellen Topologie, antwortet dieser mit einer `RING_MEMBER_ADD_REFUSE` Nachricht. Ein Knoten antwortet mit `RING_MEMBER_ADD_RESPONSE`, wenn er Teil der virtuellen Topologie ist, d. h. wenn er Mitglied oder aber auch nur ein Seitenarm des Rings ist. Teil der Nachricht sind zusätzlich die Ringnachbarn, damit der neue Knoten zwei benachbarte Knoten finden kann. Sind diese ermittelt, werden die betreffenden Ringknoten mit einer `RING_MEMBER_ADD_ACCEPT`

[7] Entfernt sich ein Nachbar, der nicht Teil der Ringstruktur ist, sind für einen Knoten keine expliziten Aktionen nötig. Da jedoch der verschwundene Knoten Nachbar eines anderen Ringknotens war, können trotzdem indirekte Maßnahmen nötig sein.

[8] Liegt nur eine Verbindung vor, kann der Knoten nur Teil der Ringstruktur, nicht aber Mitglied sein.

Nachricht und dem jeweiligen alten, zu ersetzendem Nachbarn, über die neue Ringzusammensetzung informiert. Sollten mehrere Möglichkeiten hinsichtlich der Position im Ring zur Verfügung stehen, so kann eine frei gewählt werden - ein Ring hat keine Vorzugsrichtung.

Existiert unter allen Kommunikationspartnern des neuen Knotens kein Paar, welches benachbart ist, so ist zu diesem Zeitpunkt keine Aufnahme im Ring möglich. Was bleibt, ist somit nur als Teil der Topologie einen Seitenast zu bilden. In Abbildung 4.22 ist der

```
Neuer Knoten:
    send RING_MEMBER_ADD_REQUEST to all neighbors;

    Bei Empfang von RING_MEMBER_ADD_RESPONSE;
        analyse neighbors;
        select neighbors;
        send RING_MEMBER_ADD_ACCEPT to selected neighbors;

Willkürlicher Knoten:
    Bei Empfang von RING_MEMBER_ADD_REQUEST:
        if( Knoten is element of the ring topology  )
            send RING_MEMBER_ADD_RESPONSE;
        else
            send RING_MEMBER_ADD_REFUSE;
        fi;

    Bei Empfang von RING_MEMBER_ADD_ACCEPT:
        update neighbor information;
```

Abbildung 4.22: PseudoCode für die Neuaufnahme eines Knotens in die Ringstruktur

Pseudocode für die Neuaufnahme eines Knotens in die Ringstruktur zu sehen.

Durch das Abhandenkommen eines Knoten kann die Ringstruktur im Segment geteilt werden, wenn der betreffende Knoten Teil einer Kette ist oder zwei oder mehrere Ringe miteinander verbindet. In den meisten Fällen wird in solch einer Situation nach dem Entfernen dieses Verbindungsknotens das Segment geteilt, da, wenn andere Verbindungen existieren würden, kein Verbindungsknoten notwendig ist.

Trotzdem muss versucht werden, ob die Struktur noch zusammenfügbar ist. Verantwortlich hierfür sind die Ringknoten, die durch die Trennung den Ringnachbarn verloren haben. Mit diesen Knoten muss eine Teilringinitiierung, wie zu Beginn, durchgeführt werden, um auf diese Weise den Kontakt zu anderen Ringknoten wieder herzustellen.

Es kann bei diesem Vorgang geschehen, dass der Ring nur auf einer Seite geschlossen werden kann und deshalb neben einem minimalen Ring längere Ketten in der Topologie auftreten. In der Praxis sollte dieses Verhalten auf Grund der für die Ringtopologie geforderten hohen Konnektivität nicht oder nur sehr selten auftreten.

Abschließend wird noch die Aufwändigkeit der Aufrechterhaltung der Baum- und Ringstruktur miteinander verglichen.

Hinsichtlich der Komplexität der Operationen besteht kein Unterschied. Wie die Aus-

wertungen in Kapitel 6 zeigen werden, müssen zur Aufrechterhaltung der Ringstruktur im Durchschnitt weniger Nachrichten versendet werden, als dies bei der Baumstruktur der Fall ist. Somit treten bei der Ringstruktur weniger Aufrechterhaltungsoperationen auf. Dies liegt primär daran, dass keine so hohen Anforderungen an die Ringstruktur gestellt werden wie an die Baumstruktur. Eine Baumtopologie ist streng als Baum in einem Graphen definiert. Bei der Ringtopologie ist die Definition nicht eindeutig, da neben physikalischen auch logische Ringe auftreten dürfen. Hiermit liegen gleichzeitig mehrere Alternativen vor und eine Verletzung der Vorgabe tritt weniger oft auf. Eine Überprüfung, ob die Struktur noch den Anforderungen genügt, muss seltener durchgeführt werden.

4.3.3.4 Bewertung

Die Erstellung einer virtuellen Topologie muss sich amortisieren um deren Einsatz zu rechtfertigen. Primäre Aufgabe der virtuellen Topologie ist die Reduzierung der Nachrichten, die gesendet werden müssen, um alle Knoten in einem Graph oder Segment zu erreichen. Durch Erzeugung der Topologie entsteht jedoch zunächst Mehraufwand, der sich sukzessive amortisieren muss.

Baumtopologie: Sei $|E|$ die Anzahl der Kanten und $|V|$ die Anzahl der Knoten in einem Graphen $G = (E, V)$. Um einen Baum zu erstellen sind $2|E| - |V| + 1$ Nachrichten nötig. Diese Zahl ergibt sich, da eine Nachricht über alle Kanten verschickt wird, wobei die eingehende Kante eine Ausnahme darstellt. Der Initiator versendet jedoch Nachrichten an alle ausgehenden Kanten, so dass dies durch die Addition mit 1 berücksichtigt werden muss.

Ist der Baum erstellt, sind nur noch $|V|-1$ Nachrichten nötig, um alle Knoten innerhalb des Graphen zu erreichen, während ohne zusätzliche Topologie weiterhin $2|E| - |V| + 1$ Nachrichten erforderlich sind. Zu bestimmen ist ein Amortisationsfaktor A_{Baum}, der angibt, ab wie vielen Nachrichten die Erzeugung des Baumes lohnend ist. A_{Baum} berechnet sich durch

$$\begin{aligned} A_{Baum} \cdot (|V| - 1) + 2|E| - |V| + 1 &= A_{Baum} \cdot (2|E| - |V| + 1) \\ \Rightarrow \quad A_{Baum} &= \frac{2|E| - |V| + 1}{2(|E| - |V| + 1)}. \end{aligned} \tag{4.3}$$

Gleichung 4.3 benötigt sowohl $|E|$ als auch n zur Bestimmung des Faktors A_{Baum}, was zur Folge hat, dass Werte für A_{Baum} nicht direkt offensichtlich sind. Aus diesem Grund wird nachfolgend für konkrete Graphen untersucht, wie viele Nachrichten über den Baum versendet werden müssen, damit sich die Erstellungskosten amortisieren. Als Testgraphen sollen die in Abbildung 4.23 dargestellten Beispiele dienen.

Für ein vermaschtes $a \times a$-Gitternetz mit n Knoten gilt[9]. $|E|_{Gitter} = 4|V| - 6\sqrt{|V|} + 2$.

[9] Ein vollvermaschtes $a \times b$ Gitter, wie in Abbildung 4.23a zu sehen, besteht aus

$$|E_{\text{Gitter}}| = a(b-1) + b(a-1) + 2(a-1)(b-1) = 4ab - 3(a+b) + 2, \quad (a > 1, b > 1)$$

Kanten. Für $a = b$ und $a^2 = |V|$ ergibt sich

$$|E_{\text{Gitter}}| = 4|V| - 6\sqrt{|V|} + 2, \quad |V| > 1 \tag{4.4}$$

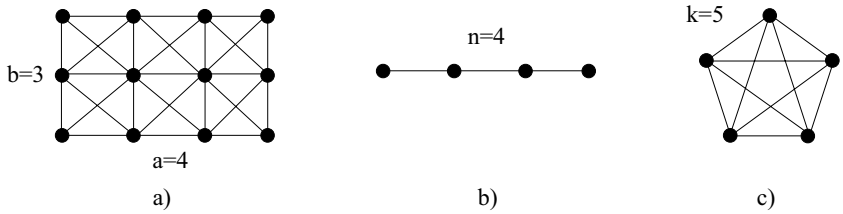


Abbildung 4.23: Verschiedene Spezialfälle für Graphen. a) zeigt ein vermaschtes $a \times b$-Gitter, b) eine Linie als entarteten Graph und c) einen vollvermaschten Graph.

Somit ergibt sich für den Amortisationsfaktor A_{Baum}^{Gitter} für ein $a \times a$ mit $|V|$ Knoten

$$A_{Baum}^{Gitter} = \frac{7|V| - 12\sqrt{|V|} + 5}{6|V| - 12\sqrt{|V|} + 6} \quad \Rightarrow \quad \lim_{|V| \to \infty} A_{Baum}^{Gitter} = \frac{7}{6} \approx 1,167. \tag{4.5}$$

Für eine lineare Kette, wie sie in Abbildung 4.23b dargestellt ist, fällt diese Kosten-Nutzen-Rechnung nicht so positiv aus. Die Mächtigkeit der Kantenmenge entspricht $|E| = |V| - 1$. Die Amortisationsrechnung ergibt

$$A_{Baum}^{Kette} = \frac{|V| - 1}{2(|V| - 1 - |V| + 1)} = \infty.$$

Dies bedeutet, die Überlagerung mit einem Baum amortisiert sich nie. Das schlechte Ergebnis resultiert daraus, dass die lineare Kette schon einen Baum darstellt. Reale und virtuelle Topologie sind somit identisch. Ein Nutzen durch die Überlagerung entsteht nicht und der Aufwand der unnützen Erzeugung kann nie beglichen werden. Liegt ein solcher Graph vor, dürfte somit keine zusätzliche Topologie erzeugt werden. Im Allgemeinen ist aber kein Wissen über die momentan existierende tatsächliche Topologie vorhanden - außer die Anwendung gibt es vor.

In einem vollvermaschten Graph (siehe Abbildung 4.23c) ist ein Knoten mit sämtlichen anderen direkt verbunden. Die Anzahl der Kanten $|E|_{vv}$ ist somit $\binom{|V|}{2} = \frac{|V|(|V|-1)}{2}$. Gleichung 4.3 liefert sodann

$$A_{Baum}^{vv} = \frac{|V| - 1}{|V| - 2} \quad \Rightarrow \quad \lim_{|V| \to \infty} A_{Baum}^{vv} = 1$$

Bei genügend großen Graphen gleichen sich die Kosten für die Erstellung somit bereits nach einer Nachricht aus. Zu beachten ist auch die schnelle Konvergenz, da bereits bei $|V| = 3$, der Amortisationsfaktor $A_{Baum}^{vv} = 2$ ist. Folglich hat sich auch bei kleinen Graphen nach bereits zwei Nachrichten die Erzeugung gelohnt. Das positive Ergebnis resultiert aus der extremen Reduzierung der benötigten Kanten.

Tabelle 4.1 vergleicht noch einmal ein $a \times a$-Gitter mit einem vollvermaschten Graphen hinsichtlich Amortisierbarkeit bei niedriger Knotenanzahl. Zu erkennen ist, dass in beiden Graphen stets mit zwei Nachrichten eine Amortisierung der Topologieerzeugung vorliegt.

Typ	A_{Baum}	$\|V\| = 4$	$\|V\| = 9$	$\|V\| = 16$	$\|V\| = 25$
$a \times a$-Gitter	$\frac{7\|V\|-12\sqrt{\|V\|}+5}{6\|V\|-12\sqrt{\|V\|}+6}$	1,50	2,33	1,28	1,25
vollvermascht	$\frac{\|V\|-1}{\|V\|-2}$	1,50	1,14	1,07	1,04

Tabelle 4.1: Amortisationsrechnung für die Baumtopologie anhand ausgewählter Graphen und Netzwerkgrößen

Welche Topologien in realen Umgebungen auftreten, hängt von der Personendichte[10] ab. Mit zunehmenden ρ_P steigt die Wahrscheinlichkeit, dass eine Teilmenge vollvermascht ist.

Ringtopologie: Wieder seien $|E|$ die Anzahl der Kanten und $|V|$ die Anzahl der Knoten in einem Graphen $G = (E, V)$. Es sind nach obiger Methode maximal $4|E|$ Nachrichten notwendig um den Ring zu erstellen, da im ersten Schritt jeder Knoten über seine Kanten die notwendigen Informationen anfordert und im zweiten Schritt die lokalen Ringinformationen über die ausgehenden Kanten ausgetauscht werden.

Die Tatsache, dass i. A. ein logischer Ring erstellt wird, erschwert die Angabe einer konkreten Nachrichtenanzahl, die notwendig ist um jeden Knoten zu erreichen. Würde ein realer Ring erzeugt, so würden $|V| - 1$ Nachrichten benötigt um eine Nachricht im Netz zu verteilen. In einem logischen Ring kann diese Zahl erheblich von diesem Optimalwert abweichen, da ein Nachfolgeknoten nicht der nächste physisch direkt erreichbare Knoten sein muss.

$$A \cdot (|V| - 1) + 4|E| = A \cdot (2|E| - |V| + 1) \quad \Rightarrow \quad A_{Ring} = \frac{|E|}{|E| - |V| + 1} \tag{4.6}$$

Um Gleichung 4.6 besser zu interpretieren und mit den Ergebnissen der Baumtopologie vergleichen zu können, werden die Beispielnetzwerke, die in Abbildung 4.23 zu sehen sind, verwendet.

Für ein vollvermaschtes $a \times b$-Gitternetz belaufen sich die Amortisationskosten gemäß Gleichung 4.6 auf

$$A_{Ring}^{Gitter} = \frac{4(2|V| - 3\sqrt{|V|} + 1)}{3(|V| - \sqrt{|V|} + 1)} \quad \Rightarrow \quad \lim_{|V| \to \infty} A_{Ring}^{Gitter} = \frac{8}{3}. \tag{4.7}$$

Für eine lineare Kette fällt ebenso wie bereits für die Baumtopologie die Amortisationsrechnung negativ aus.

$$A_{Ring}^{Kette} = \frac{4|V| - 4}{2(|V| - 1 - |V| + 1)} = \infty$$

Eine Amortisation tritt nicht ein, da wie im obigen Fall reale und virtuelle Topologie quasi identisch sind. Die Struktur der Kette ist zu einfach und macht eine zusätzliche Ebene nicht nötig.

[10] Personendichte $\rho_P = \frac{\text{Anzahl an Personen}}{\text{Fläche}}$

Bleibt zuletzt noch die Auswertung für ein vollvermaschtes Netzwerk. Gleichung 4.6 liefert sodann

$$A^{vv}_{Ring} = \frac{2|V|^2 - 2}{|V|^2 - 2|V| + 2} \quad \Rightarrow \quad \lim_{|V|\to\infty} A^{vv}_{Ring} = 2. \tag{4.8}$$

Zu bemerken ist, dass die Grundlage für die Amortisationsrechnung der obig erwähnte Ringerzeugungsalgorithmus ist. Für einen vollvermaschten Graphen existieren Verfahren, die mit $n-1$ Nachrichten einen Ring erzeugen[11]. Wenn jedoch bekannt wäre, dass ein vollvermaschter Graph vorliegt, würde generell keine virtuelle Topologie benötigt, da mit $|V|-1$ Nachrichten alle restlichen Teilnehmer erreicht werden können.

Diese vollvermaschten Netzwerke sind im Fall einer vorliegenden ad hoc Vernetzung häufiger als vermutet wird. Liegen abgegrenzte Flächen vor, deren Ausdehnung sich unterhalb des Kommunikationsradius beläuft, wie es beispielsweise in U-Bahnhöfen oder Flugzeugsteigen der Fall ist, ist ein vollvermaschtes Netzwerk oft die reale Struktur.

In Tabelle 4.2 sind für das Gitter und den vollvermaschten Graphen ausgewählte Werte

Typ	A_{Baum}	$\|V\| = 4$	$\|V\| = 9$	$\|V\| = 16$	$\|V\| = 25$
$a \times a$-Gitter	$\frac{4(2\|V\|-3\sqrt{\|V\|}+1)}{3(\|V\|-\sqrt{\|V\|}+1)}$	4	3.33	3.11	3
vollvermascht	$\frac{2\|V\|^2-2}{\|V\|^2-2\|V\|+2}$	4	2.86	2.46	2.33

Tabelle 4.2: Amortisationsrechnung für die Ringtopologie anhand ausgewählter Graphen und Netzwerkgrößen

für die Amortisation angegeben.

Abschließend kann gesagt werden, dass sich eine zusätzliche Ringstruktur mit zunehmender Kantenanzahl schneller amortisiert.

4.3.4 Gruppenbildung

Nachdem in den vorherigen Abschnitten die Basis für die Gruppenbildung gelegt wurde, wird im Folgenden die eigentliche Erzeugung von Gruppen erläutert. Bereits in Abschnitt 1.1.1 wurde erwähnt, dass es mehrere Gründe gibt, warum Gruppen gebildet werden. Entweder Objekte schließen sich zu Gruppen zusammen, weil sie für sich alleine nicht in der Lage sind ein Ziel zu erreichen oder Gruppen bilden sich, da sich ein Mehrwert für jeden einzelnen ergibt.

Eine erste Variante wird im folgenden Text als ähnlichkeitsbasierte (similarity based) Gruppenbildung bezeichnet. Die Namensgebung rührt daher, dass „ähnliche“ Profileinträge die Parameter für die Gruppierung sind.

Mit nutzenbasiertem (benefit based) Gruppieren wird eine zweite Variante bezeichnet, da eine Gruppe nur gebildet wird, wenn alle Gruppenteilnehmer einen Vorteil erkennen können. Für beide Gruppierungsarten wird ein Algorithmus vorgestellt.

Beiden Gruppierungsarten ist gemein, dass der Gruppierungsprozess mehrstufig (siehe hierzu auch Abschnitt 3.3.1.3) organisiert ist. In einem ersten Schritt erzeugt jeder Knoten

[11] Beispielalgorithmus: Die erste Nachricht wird an einen willkürlichen Teilnehmer gesendet. Jeder Empfänger trägt sich in eine der Nachricht anhängenden Liste ein und leitet sie an einen Teilnehmer weiter, der noch nicht in der Liste steht.

eines Segments seine individuelle *Lokale Gruppe*. Diese besteht aus einer Teilmenge der direkten Nachbarn eines Knotens. In der *Lokalen Gruppe* sind die Nachbarn enthalten, mit denen ein Knoten eine Gruppe bilden will, d. h. das sind diejenigen, die sich auf Grund der Ähnlichkeit oder eines potenziellem Nutzens zur Gruppenbildung eignen. Der erste Schritt hat somit lokalen selektiven Charakter. Die lokale Gruppe stellt damit den individuellen Ausgangspunkt der weiteren globalen Gruppierung dar.

Im zweiten Schritt werden die *Lokalen Gruppen* ausgetauscht, um auf diese Weise eine bzw. auch mehrere, jedoch im Segment eindeutige *Globale Gruppen* zu erzeugen, die schließlich das Ergebnis des Gruppierungsprozesses darstellen.

Der Gruppierungsprozess wird somit in zwei Schritten durchgeführt, da auf diese Weise die Rechenleistung der mobilen Geräte besser ausgenutzt wird. So muss jedes Gerät eine lokale Gruppe bilden, die jedoch nur aus einem Teil der Segmentmitglieder besteht. Für die Bildung der globalen Gruppe muss nur noch die Kombination der lokalen Gruppen bestimmt werden.

4.3.4.1 Formale Definitionen

Bevor das Vorgehen der Gruppenbildung im Detail algorithmisch erklärt wird, ist eine formale Definition des Gruppenbegriffs notwendig. Hierzu müssen zuvor mehrere andere Begrifflichkeiten einzuführen.

Die Basis für die Gruppenbildung stellt ein Profil dar. Als Definition eines Profils soll die Definition 2 in Abschnitt 3.2.1 dienen.

Im Folgenden sei stets $G = (V, E)$ ein Graph, mit $V = \{v_1, v_2, \ldots, v_n\}$ und $E \subseteq V \times V$ mit $(v_i, v_j) \in E$, wenn die Distanz $d(v_i, v_j) \leqslant r_K$ und eine Menge $\mathcal{P}$ an Profilen mit $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ sind.

Um Profile vergleichen zu können, sei auf $\Pi = \Pi_1 \times \ldots \times \Pi_m$ durch eine Distanzfunktion ρ (Beispiele für Distanzfunktionen wurden in Abschnitt 3.3.1.1 eingeführt) eine Metrik definiert. Hierdurch wird Π zum *metrischen Raum*.

Bisher existiert keine Verbindung zwischen einem Knoten im Graphen $G = (V, E)$ und den Profilen der Knoten. Deshalb existiert eine Funktion ϕ: $V \longrightarrow \mathcal{P}$, welche jedem Knoten $v_i \in V$ ein Profil P_j zuordnet, d. h. $\phi(v_i) = P_j$. Die Umkehrfunktion von ϕ sei mit ϕ^{-1} bezeichnet. ϕ^{-1} existiert, wenn es kein v_i und v_j gibt, mit $\phi(v_i) = \phi(v_j) = P_k$. Gewährleistet wird dies durch die eindeutige ID im Benutzerprofil. ϕ^{-1} wird benötigt, um bei einem gegebenen Profil auf den assoziierten Knoten schließen zu können.

Für die Gruppierung muss ein Mechanismus existieren, der anhand einer bereits existierenden Gruppe und eines neuen Teilnehmers entscheidet, ob dieser neuen Teilnehmer in die Gruppe mit aufgenommen werden kann.

Definition 11: *Entscheidungsoperator* ϵ
Sei $\mathfrak{P}(V)$ die Potenzmenge der Knotenmenge V des Graphen G. Dann sei ϵ definiert als, $\epsilon : \mathfrak{P}(V) \times V \rightarrow \{\texttt{true, false}\}$.

□

Der Entscheidungsoperator ϵ muss jeweils für die ähnlichkeits- und nutzenbasierte Gruppierung definiert werden.

Für den Fall der nutzenbasierten Gruppenbildung ist der Begriff der Paretoeffzienz von

zentraler Bedeutung

Definition 12: *Paretoeffzienz*
Eine Situation wird als *Pareto-effzient* oder *Pareto-optimal* charakterisiert, falls es nicht möglich ist, ein Individuum besser zu stellen ohne ein anderes Individuum schlechter zu stellen. Eine Situationsänderung stellt eine Pareto-Verbesserung dar, falls sie ein Individuum besser stellt ohne ein anderes schlechter zu stellen [BEG02; HI93]. Eine Pareto-optimale Situation liegt vor, falls keine Pareto-Verbesserung möglich ist.

□

Wird der Begriff der Paretoeffzienz auf den Bereich der Gruppenbildung angewendet, so bedeutet dies, dass durch die Gruppe seitens ihrer Teilnehmer für jeden eine Pareto-Verbesserung eintreten muss. Ziel der Gruppenbildung sollte es sein, Pareto-optimale Gruppen zu erzeugen. Dies kann jedoch im MoPiDiG Framework nicht immer garantiert werden, da dies hinsichtlich der Komplexität und Verarbeitungszeit in einem dynamischen Umfeld, wie es in MoPiDiG vorherrschend ist, nicht immer möglich ist. Die Tendenz nicht-Pareto-optimaler Gruppen nimmt mit der Gruppengröße zu. Entscheidend für das MoPiDiG Framework ist jedoch, dass die Alternativlösung nicht sehr stark von der besten Lösung abweicht, wie in Kapitel 6 gezeigt wird.

Um eine Bewertung des Gruppenwertes vorzunehmen, dient eine Payoff Funktion π.

Definition 13: *Payoff Funktion π*
Die Payoff Funktion $\pi : V^k \rightarrow \mathbb{R}$ weist jeder Gruppe $\mathcal{G}$ einen Wert zu, welcher dem Nutzen der Gruppe entspricht. Damit neue Elemente in die Gruppe mit aufgenommen werden, muss sich durch deren Aufnahme π erhöhen[12], da ansonsten die Bildung einer Gruppe nicht ausreichend motiviert ist.

□

Soll das Profil P_v eines Knotens v zu einer Gruppe G hinzugefügt werden und sei $\mathfrak{P}(\mathcal{G})$ die Potenzmenge der lokalen Gruppe $\mathcal{G}$, müssen zwei Bedingungen erfüllt sein:

$$\text{Notwendige Bedingung:} \quad \pi(G \cup P_v) > \pi(G) \tag{4.9}$$

$$\text{Hinreichende Bedingung:} \quad \forall G_i \in \mathfrak{P}(G) : \pi(G \cup P_v) > \pi(G_i) \tag{4.10}$$

Ferner gilt für alle Teilmengen A und B von G mit $A \subseteq B$: $\pi(A) \leq \pi(B)$.

Gleichung 4.10 fordert, dass die Payoff Funktion π all ihrer Teilmengen kleiner ist als für die gesamte Gruppe. Die Gleichungen 4.9 und 4.10 definieren somit den Entscheidungsoperator ϵ aus Definition 4.3.4.1 für das nutzenbasierte Gruppieren.

Für das ähnlichkeitsbasierte Gruppieren muss für den Entscheidungoperator ϵ der Begriff der Ähnlichkeit spezifiziert werden.

Definition 14: *Ähnlichkeit* $\triangleq$
Zwei Objekte A und B sind als ähnlich anzusehen, wenn gemäß einer Distanzfunktion $d(A, B)$ ein definierter oberer Schwellwert d_{MAX} nicht überschritten ist (vgl. hierzu Distanz- und Ähnlichkeitsmaße in Abschnitt 3.3.1.1). Hierdurch wird eine Relation $A \triangleq B \subseteq \Pi \times \Pi$ definiert.

[12] Alternativ zum Nutzen einer Gruppe könnten auch die Kosten betrachtet werden, die sich mit und ohne Gruppe ergeben. Im Gegensatz zum Nutzen müssen sich die Kosten jedoch durch die Gruppenbildung reduzieren. Für manche Szenarien ist ein Nutzen und für andere sind die Kosten offensichtlicher. Im Allgemeinen können jedoch beide Varianten ineinander überführt werden.

Eigenschaften der Relation $\triangleq$:

$$\begin{aligned} \text{Reflexivität:} \quad & A \triangleq A \\ \text{Symmetrie:} \quad & A \triangleq B \Rightarrow B \triangleq A \\ \text{Intransitivität:} \quad & A \triangleq B \ \wedge \ B \triangleq C \nRightarrow A \triangleq C \end{aligned}$$

□

Die Reflexivität ist erfüllt, da Gleichheit vorliegt und dies eine Distanz $d(A, A)$ von Null impliziert und somit stets kleiner als d_{MAX} ist. Die Symmetrieeigenschaft muss nicht gegeben sein, da unterschiedliche Schwellwerte d_{MAX} vorliegen können. Im Folgenden wird jedoch postuliert, dass für jeden Anwendungsbereich d_{MAX} definiert wird und somit die Symmetrieeigenschaft erfüllt ist. Die Intransitivität ist in Abbildung 4.24 ersichtlich, wo

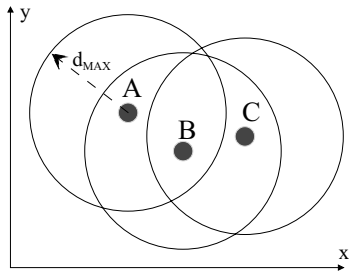


Abbildung 4.24: Intransitivität der Ähnlichkeitsrelation

zwar $A \triangleq B$ und $B \triangleq C$ aber $A \nriangleq C$.

Die Relation $\triangleq$ ist somit keine Ähnlichkeitsrelation (reflexiv, symmetrisch, transitiv) im mathematischen Sinne. Hieraus resultiert, dass eine zu gruppierende Menge M an Teilnehmern nicht in disjunkte Partitionen aufgeteilt können werden muss. Deshalb müssen Methoden verwendet werden, die eine Gruppeneinteilung bewerkstelligen, unter der Voraussetzung, dass Gruppen hinsichtlich der Ähnlichkeit der Teilnehmer optimal sind.

Der Entscheidungsoperator ϵ liefert somit bei der ähnlichkeitsbasierten Gruppenbildung genau für die Elemente den Wert `true`, wenn sie Element der Relation $\triangleq$ sind.

4.3.4.2 Lokale Gruppenbildung

Der Gruppierungsprozess ist mehrstufig organisiert und besteht aus zwei Schritten. Als erstes wird eine *Lokale Gruppe* erzeugt.

Definition 15: *Lokale Gruppe* $\mathcal{G}$
Sei N_v^1 die Menge an Kommunikationspartnern eines willkürlichen Knotens v_i über q Kanten, $\mathfrak{P}(N_v^1)$ deren Potenzmenge und sei $K = \left\{M \in \mathfrak{P}(N_v^1) \,|\, \epsilon(M, v_i) = \texttt{true}\right\}$. Dann ist $\mathcal{G} \subseteq N_{v_i}^1$:

$$\mathcal{G} = \max_{\forall x \in K} |x| \tag{4.11}$$

Die Menge aller n lokalen Gruppen $\mathcal{G}$, die im Segment S existieren, sei bezeichnet mit $\mathfrak{G}$.
□

Eine lokale Gruppe $\mathcal{G}$ eines Knotens v ist somit die Teilmenge der direkten Nachbarn von v mit der höchsten Kardinalität, welche gemäß des Entscheidungsoperators ϵ zu einer Gruppe zusammengefasst werden können. Zu bemerken ist, dass diese allgemeine Definition die Anzahl der Mitglieder einer lokalen Gruppe nicht einschränkt und somit nicht von vorhandenen Gruppendefinitionen (siehe Abschnitt 4.4.3.2) abhängt. Die lokale Gruppe beschränkt sich auch nicht nur auf die direkten Nachbarn, die einer virtuellen Topologie angehören, sondern bezieht stets sämtliche direkten Nachbarn mit ein.

Jeder Knoten eines Segments führt diesen Schritt durch, so dass - wenn ein Segment aus n Knoten besteht - n lokale Gruppen vorliegen.

Um eine lokale Gruppe zu erzeugen ist somit zuerst die Beschaffung der notwendigen Profilinformation nötig, um anschließend das Maximum der Gruppenteilnehmer eruieren zu können. Der erste Teil ist unabhängig von der Gruppierungsart, wogegen die Selektion der Gruppenteilnehmer aus den direkten Nachbarn für die nutzen- und ähnlichkeitsbasierte Gruppenbildung unterschiedlich durchgeführt wird.

Anforderung der Profilinformation: In diesem Vorgang werden die direkten Nachbarn aufgefordert Profildaten an einen Knoten zu senden. Da ein Benutzerprofil eine Vielzahl an Einträgen aufweisen kann, muss deshalb zuerst der Teil des Profils bestimmt werden, der zur Gruppenbildung notwendig ist. Definiert ist dieser jedoch bereits in der Gruppenbeschreibung (siehe Abschnitt 4.4.3.2). Somit muss nicht dass komplette Profil ausgetauscht werden, was sowohl aus Ressourcen- als auch aus Sicherheitsgründen positiv zu bewerten ist. Dieses Verfahren ähnelt dem der Anonymitätstemplates von Astor [Ast02], welches bereits in Abschnitt 3.2.5 erläutert wurde.

Will ein Knoten v die Profildaten anfordern, sendet er eine `PROFIL_DATA_QUERY` Nachricht mit der Beschreibung der nötigen Profilinformationen an seine Nachbarn. Diese senden eine `PROFIL_DATA_INFORM` Nachricht zurück, wenn sie daran interessiert sind ein Gruppenmitglied zu werden. Im anderen Fall wird mit `PROFIL_DATA_REFUSE` angezeigt, dass die notwendigen Daten nicht übermittelt werden. In Abbildung 4.25 ist der Pseudocode der

```
Initiatorknoten:
    send von PROFIL_DATA_QUERY to all neighbors;

Bei Empfang von PROFIL_DATA_QUERY:
    sender = getSender();
    if( permission == ok )
       send PROFIL_DATA_INFORM with Profileinformation to sender;
    else
       send PROFIL_DATA_REFUDSE to sender;
    fi;
```

Abbildung 4.25: Pseudocode für die Anforderung von Profilinformation

Profilanforderung angegeben.

Über die Frage, ob ein Gerät an einer Gruppierung teilnimmt, entscheidet die dem Gerät assoziierte Person. Nur wenn dies erwünscht ist, werden die notwendigen Aktionen

veranlasst. Würden Teilnehmer implizit, d. h. ohne deren Intention zu kennen, in die Gruppenbildung mit eingeschlossen und erst am Ende um deren explizite Teilnahme befragt, würde eine negative Antwort einer Person eine erneute Gruppenbildung erfordern, wenn sich der Nutzen nicht verschlechtern soll.

In Abbildung 4.26, Abbildung 4.26a ist zu sehen, wie Knoten 7 `PROFIL_DATA_QUERY`

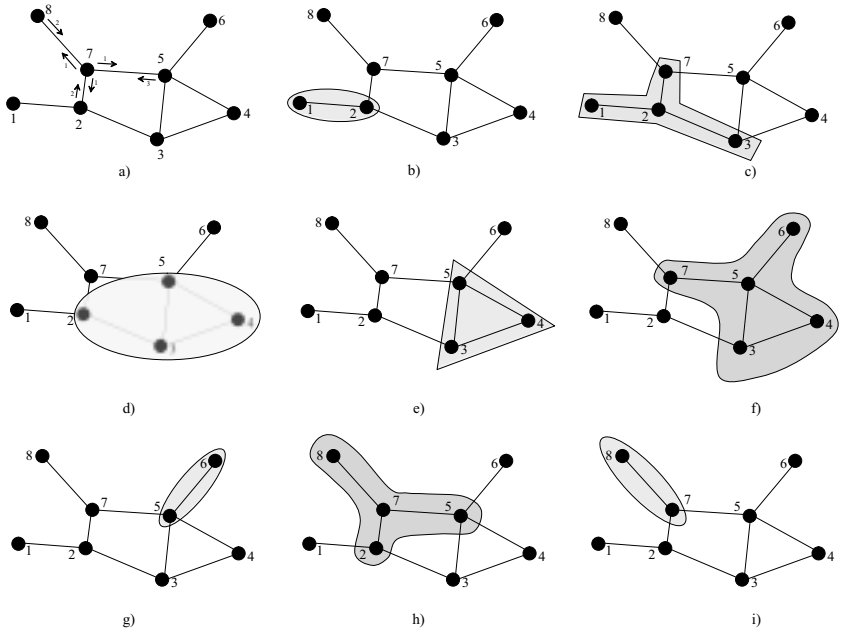


Abbildung 4.26: a) zeigt den Austausch der Nachrichten zum Erhalt der lokalen Gruppe. Mit der Nummer eins werden `PROFIL_DATA_QUERY` Nachrichten angedeutet. Nummer zwei steht für `PROFIL_DATA_INFORM` und Nummer drei für `PROFIL_DATA_REFUSE` Nachrichten. In b)-h) sind sämtliche Kommunikationspartner, die für je eine lokale Gruppe herangezogen werden, ersichtlich.

Nachrichten aussendet. Knoten 2 und 8 deuten mit `PROFIL_DATA_INFORM` an, an der Gruppierung teilzunehmen. Knoten 5 zeigt sein Desinteresse mit `PROFIL_DATA_REFUSE`.

Die Abbildungen 4.26b-4.26h zeigen, welche Knoten jeweils für die lokale Gruppenbildung herangezogen werden. So muss Knoten 8 nur entscheiden, ob Knoten 7 noch zur lokalen Gruppen addiert werden soll. Knoten 5 muss diesen Vorgang hingegen für die Knoten 3, 4, 6 und 7 entscheiden.

Bildung der lokalen Gruppe aus den gesammelten Profilinformationen: Nachdem die notwendigen Daten auf allen Knoten vorliegen, kann aus diesen eine lokale Gruppe erzeugt werden. Im Nachfolgenden wird das Vorgehen jeweils für die nutzen- und ähnlichkeitsbasierte Gruppenbildung erläutert.

Nutzenbasierte Gruppenbildung: Die nutzenbasierte Gruppenbildung ist abhängig vom Entscheidungsoperator ϵ, der in diesem Fall nur durch die Payoff-Funktion π

definiert wird.

Aufgabe des lokalen Gruppierens ist es, die Teilmenge aus den direkten Nachbarn zu selektieren, deren Profileinträge π zu maximieren.

Die Modellierung der Profilpunkte und das Eruieren des Maximums erfolgt mit Hilfe eines *Profilnetzwerks*. In Abbildung 4.27a repräsentieren die Punkte in dem Koordina-

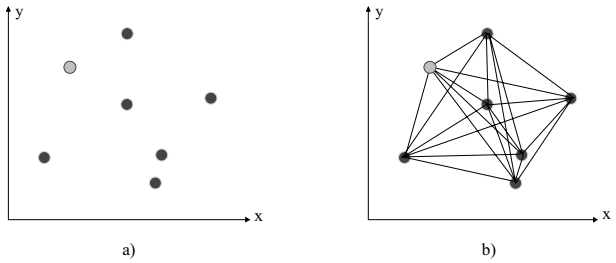


Abbildung 4.27: Darstellung der Profile im zweidimensionalen Koordinatensystem (a). Modellierung des Profils als vollständiger Graph (b)

tensystem Profilpunkte. Hierbei entspricht dem grauen Punkt das Profil des Teilnehmers, der die lokale Gruppe bestimmt und die restlichen Punkte entsprechen den Profilen seiner direkten Nachbarn.

In der Abbildung 4.27b ist die Modellierung der Profilinformationen als vollständiger Graph dargestellt. Die Kanten im Graph sind gewichtet. Die Gewichte w_i entsprechen der Änderung von π, die sich an einer Kante ergibt.

Das Finden derjenigen Profileinträge, die π maximieren, kann somit auf das Problem des Findens des längsten Weges in einem gewichteten Graphen zurückgeführt werden. Prinzipiell existieren hierfür bereits viele Verfahren[13], die z. B. in Cormen *et al.* [CLRS01] aufgeführt sind. Durch eventuell vorherrschende Eigenschaften des Entscheidungsoperators können diese jedoch nicht angewandt werden.

Der Entscheidungoperator ϵ muss - wie im folgenden Beispiel ersichtlich - nicht assoziativ sein:

$$\begin{aligned}
\pi(V_0) = &\quad \pi_0 && V_0 \in \mathcal{G} \quad \text{(Start)} \\
\pi(V_0 \cup V_1) = &\quad \pi_1 < \pi_0 \Rightarrow && V_1 \notin \mathcal{G} \\
\pi(V_0 \cup V_2) = &\quad \pi_2 > \pi_0 \Rightarrow && V_2 \in \mathcal{G} \\
\pi(V_0 \cup V_2 \cup V_1) = &\quad \pi_3 > \pi_2 \Rightarrow && V_1 \in \mathcal{G}
\end{aligned}$$

Im Beispiel wird der Knoten V_1 nicht alleine zusammen mit V_0 zu einer Gruppe zusammengefügt. Nachdem jedoch V_2 zur (lokalen Gruppe) hinzugefügt wurde, vergrößert die Hinzunahme von V_1 die Payoff Funktion, so dass auch schließlich V_1 in die (lokale) Gruppe mit integriert wird.

Hieraus folgt, dass in nicht assoziativen Fällen, um ein optimales Ergebnis zu erreichen, sämtliche Möglichkeiten der Gruppenzusammensetzung getestet werden müssen. Hat ein

[13] Es existieren Verfahren, die den kürzesten Pfad in einem Graphen ermitteln. Diese können jedoch für den Fall, dass keine Zyklen existieren, auch für den längsten Pfad Verwendung finden.

Knoten n direkte Nachbarn, müssten somit sämtliche $n!$ Permutationen überprüft werden, um die optimale Zusammensetzung zu eruieren. Dies berücksichtigt jedoch nur die Gruppen mit $n+1$ Teilnehmern. Prinzipiell sind auch kleinere Gruppen mit einer Anzahl $A : 1 \leq A \leq n$ möglich. Somit erhöht sich die Gesamtanzahl an zu testenden Kombinationen auf $\sum_{i=1}^{n} i!$.

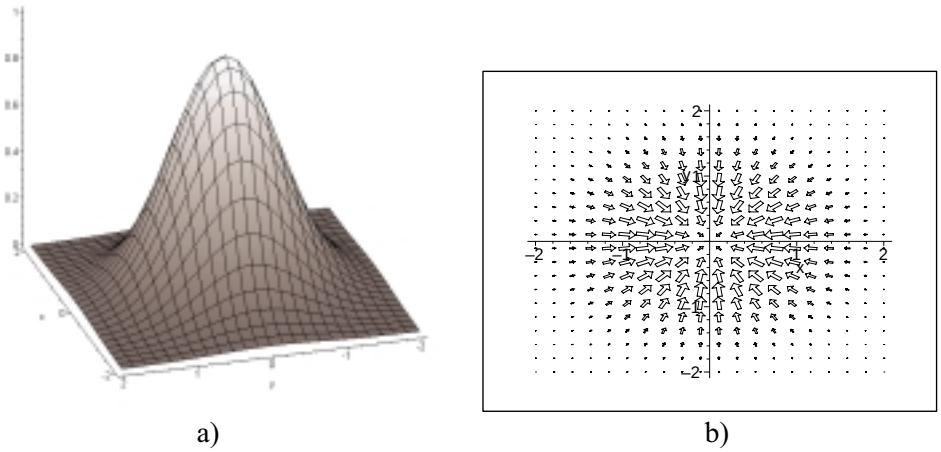
Es ist nur für eine kleine Anzahl an direkten Nachbarn möglich sämtliche Möglichkeiten zu testen. Bei fünf direkten Nachbarn müssten 153 verschiedene Möglichkeiten getestet werden. Bei acht Nachbarn steigt der Wert bereits auf über 46 000. Das ist jedoch zeitlich nicht machbar, da es sich zum einem um mobile Endgeräte handelt, deren Leistung nur begrenzt ist und zum anderen die Mobilität nicht genügend Zeit lässt, da sich Kommunikationspartner voneinander entfernen.

Um deshalb die lokale Gruppe effizienter bestimmen zu können, werden nachfolgend Heuristiken vorgestellt. Diese Heuristiken zielen entweder darauf ab, anstatt eines globalen Maximum von π ein adäquates lokales Maximum zu finden oder sie schränken die Anzahl an zu testenden direkten Nachbarn ein.

Gradienten-Methode: Die Gradienten-Methode versucht die Anzahl an Gruppierungsobjekten einzuschränken, indem zuerst in Richtung des stärksten Anstiegs der Funktion π nach möglichen Gruppierungsobjekten gesucht wird. Hierzu kann der Gradient von π verwendet werden.

$$\nabla\pi(\mathbf{x}) = \begin{pmatrix} \frac{\partial}{\partial x_1}\pi(\mathbf{x}) \\ \frac{\partial}{\partial x_2}\pi(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_n}\pi(\mathbf{x}) \end{pmatrix}$$

$\nabla f(\mathbf{x})$ ist ein Vektor, der genau in Richtung des stärksten Anstiegs zeigt. Abbildung 4.28 zeigt in Abbildung 4.28a eine Funktion f und in Abbildung 4.28b die dazugehörigen



a) b)

Abbildung 4.28: Funktion $\mathbb{R}^2 \rightarrow \mathbb{R}$ (a) mit zugehörigem Gradienten-Vektorfeld (b). Je dicker die Pfeile in b), desto größer der Anstieg der Funktion.

Gradienten-Vektoren.

Ein Gebiet, in welchem vorrangig nach potenziellen Gruppierungsobjekten gesucht wird, könnte erhalten werden, indem der Gradient eines Startpunktes berechnet wird und in Richtung des erhaltenen Vektors nach Punkten gesucht wird. Das Gebiet kann auch durch Intervallangaben (z. B. $I_x = [5..10], I_y = [3..11], I_z = [7..9]$) spezifiziert werden.

Die Konzentration auf Punkte, welche sich in Richtung des stärksten Anstiegs befinden, muss in keinem Fall zu einem globalen Maximum führen, sondern es wird i. A. ein lokales Maximum gefunden. Ein Verfahren zur expliziten Bestimmung des globalen Maximums existiert nicht. Es müssten sämtliche lokale Maxima festgestellt werden, um anschließend hieraus das globale Maximum zu bestimmen.

Die Problematik dieses Verfahren ist der zusätzliche mathematische Aufwand, der notwendig ist um den Gradienten zu berechnen. Da die Funktion π domänenabhängig ist, müsste ein Verfahren (i. A. numerischer Art) entwickelt werden, welches für eine gegebene Funktion den entsprechenden Gradienten eruiert.

Kann der Gradient von π nicht berechnet werden, kann die Richtung des stärksten Anstiegs auch mittels Suche eruiert werden. Es wird in der Umgebung des Punktes die Funktion π betrachtet und auf diese Weise wird der Bereich ermittelt, in dem π am schnellsten ansteigt.

Diese Variante der Gebietsbestimmung kommt ohne die schwierige Gradienten Berechnung aus. Doch stattdessen müssen viele Punkte berechnet werden um das Gebiet zu bestimmen. Die Anzahl der Punkte hängt von der Anzahl der betrachteten Dimensionen ab. Erfolgt eine Suche im $\mathbb{R}^n$, so sind mindestens $2 \cdot n$ Richtungen zu durchsuchen (jeweils in beide Richtungen der Achsen). Dies wäre durchaus ein vertretbarer Aufwand. Doch in den seltensten Fällen wird es ausreichen nur die Achsen zu durchsuchen. Es müssen darüber hinaus Zwischenwerte betrachtet werden. Aus diesem Grund sollten die Teilmengen des Profils, die als Grundlage für die Gruppierung dienen, nur aus wenigen Werten bestehen. Dies bedeutet nicht, dass das Gesamtprofil eine geringe Anzahl an Einträgen aufweist, sondern dass für eine Gruppe nur Teile aus dem Gesamtprofil bedeutsam sein sollen.

Gebietsdefinition: Als Abwandlung zur Gradientenmethode kann auch direkt ein Gebiet definiert werden, in welchem vorrangig nach entsprechenden Gruppierungsobjekten gesucht wird. In Abbildung 4.29a ist ein Profilnetzwerk zu sehen. Der graue Punkt repräsentiert das Profil des Teilnehmers, der die lokale Gruppe bilden möchte. Gemäß Abbildung 4.29a sind sechs direkte Nachbarn vorhanden. In Abbildung 4.29b wird als Beispiel ein Kreis als Gebiet vorgegeben, in welchem die Suche ablaufen soll. Hierdurch wird, wie in Abbildung 4.29c zu sehen ist, die Anzahl auf vier reduziert.

Ob ein Bereich, wie ein Kreis oder eine Ellipse, direkt angegeben werden kann, hängt von der jeweiligen Anwendungsdomäne ab. In Abschnitt 5.4.3.1 wird an einem konkreten Beispiel verdeutlicht, wie ein Gebiet bestimmt werden kann.

Lokales-Maximum: Eine weitere Möglichkeit ist es, nicht das globale Maximum von π zu eruieren, sondern sich auf lokale Maxima zu konzentrieren.

Um dies zu erreichen, wird bei der Lokales-Maximum-Heuristik stets der Knoten als nächster Knoten gewählt, der π maximiert. In Abbildung 4.30 ist dies am Beispiel darge-

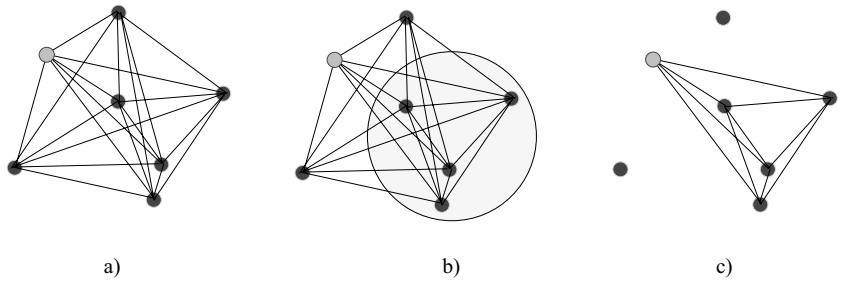


Abbildung 4.29: Reduktion der Anzahl der Gruppierungsobjekte durch Bereichsdefinition

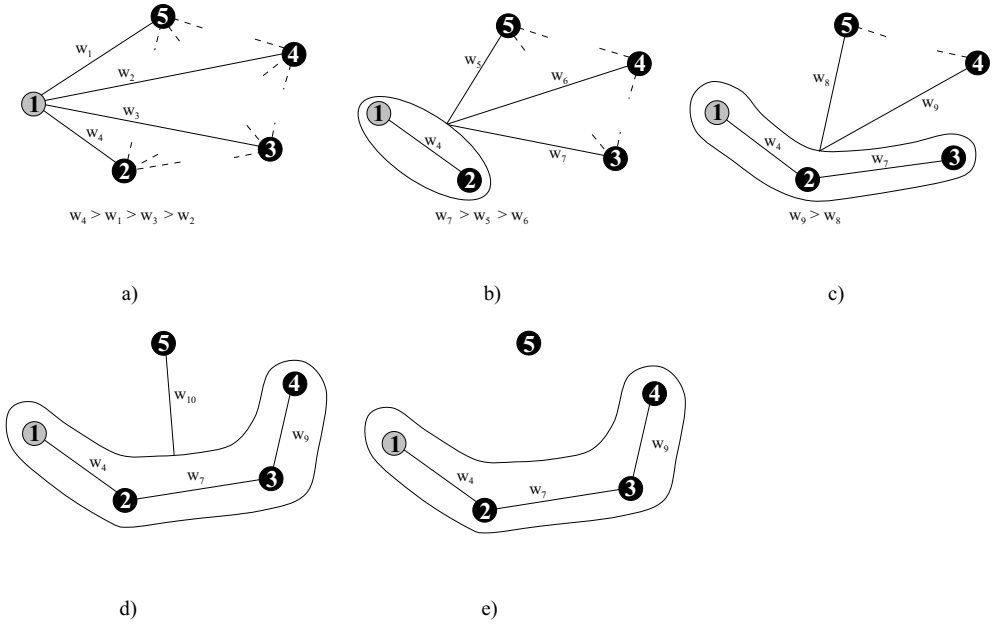


Abbildung 4.30: Lokales-Maximum-Heuristik

stellt. In Abbildung 4.30a existieren vier Möglichkeiten einen nächsten Knoten zu wählen. Gemäß dieser Heuristik wird der Knoten gewählt, dessen Gewicht w_i am größten ist, sofern es in diesem Fall π erhöht. Dies ist für den Knoten, der über die Kante mit dem Gewicht w_4 verbunden ist, gegeben. In Abbildung 4.30b und Abbildung 4.30c wird ebenso verfahren. In Abbildung 4.30d wird kein Punkt hinzugefügt, da für diesen Fall π nicht weiter maximiert werden kann. Somit zeigt Abbildung 4.30e die lokale Gruppe, welche aus zusätzlichen drei Punkten besteht.

Durch diese Methode werden die bei einer Anzahl n an direkten Nachbarn nötigen $n!$ bzw. $\sum_{i=1}^{n} i!$ Möglichkeiten auf n reduziert.

Der Algorithmus zur lokalen Gruppenbestimmung für diese Heuristik ist in Abbildung 4.31 dargestellt.

```
/* Initialisierung */
firstReferenceNode := currentPoint;
secondReferenceNode := currentPoint;
nextPoint := null;
localGroup := currentPoint;

/* Profit des Initiators */
currentProfit = profit_function( localGroup );

/* Erzeugen der lokalen Gruppe */
while((nextPoint:=getNearestProfilePoint(firstReferenceNode)) != null)
  futureProfit := profit_function( localGroup + nextPoint );
  if( futureProfit > currentProfit ) then
    localGroup          += nextPoint;
    neighbors           -= nextPoint;
    currentProfit        = futureProfit;
    firstReferenceNode  := secondReferenceNode;
    secondReferenceNode := nextPoint;
  fi;
elihw;
```

Abbildung 4.31: Pseudocode des Gruppierens bei der Lokales-Maximum-Heuristik

Diese Heuristik liefert i. A. kein globales Maximum, da es auch Wege im Profilnetzwerk geben kann, die begonnen mit einer nicht optimalen Kante schließlich π maximieren. Wie stark sich die Ergebnisse (globales Maximum vs. Lokales-Maximum-Heuristik) unterscheiden, wird in Kapitel 6 untersucht.

Dieses Verfahren kann noch abgeändert werden um nicht zu restriktiv mit der Knotenreduzierung umzugehen. Im obigen Verfahren wird nur jeweils der Knoten im Profilnetzwerk ausgewählt, der π lokal maximiert. Alternativ können auch die k besten Knoten in Betracht gezogen werden um auf diese Weise das Ergebnis zu verbessern. In diesem Fall müssen bei n direkten Nachbarn $(k-1)! \cdot k^{n-k+1}$ Fälle getestet werden. Das ist zwar kleiner als $n!$, steigt jedoch auch exponentiell an.

Ähnlichkeitsbasierte Gruppenbildung: Bei der ähnlichkeitsbasierten Gruppenbildung kann zwischen zwei Ausprägungen unterschieden werden. In einem Fall (nachfolgend expliziter Fall genannt) sind die Merkmale bekannt, die eine Gruppe bestimmen, im anderen Fall (impliziter Fall) sind diese nicht bekannt. Am Beispiel der personalisierten Museumsführung können beide Fälle erklärt werden. Im expliziten Fall soll eine Gruppe gefunden werden, deren Teilnehmer sich explizit für „impressionistische Künstler“ interessieren. Im impliziten Fall soll eine Menge an Personen anhand der vorhandenen Profileinträge so in Gruppen eingeteilt werden, dass die entstandenen Gruppen hinsichtlich ihrer Ähnlichkeit optimal sind. Dieser Vorgang entspricht dem einer Klassifikation anhand der

Profileinträge.

Expliziter Fall: Der explizite Fall benötigt für die Beantwortung, ob gewisse Einträge im Profil vorhanden sind, eine Anfragemöglichkeit an das Benutzerprofil und kann aus diesem Grund effizient realisiert werden.

Mit dem Profilanfrage-Modul, welches in Abschnitt 4.4.2.1 beschrieben ist, ist dies möglich.

Eine Gruppierungsbedingung kann aus einer Konkatenation von Profileinträgen bestehen. Ferner kann zusätzlich in der Gruppendefinition angegeben werden, ob sämtliche Angaben erfüllt werden müssen oder ob auch Teilmengen erlaubt sind. Für den letzten Fall muss folglich noch die Anzahl an passenden Einträgen eruiert werden. Wurde dieser Vorgang für alle Knoten in der direkten Nachbarschaft durchgeführt, werden sie noch gemäß ihrer Güte sortiert. Je mehr Anforderungen ein Nachbarprofil erfüllt, desto besser wird das jeweilige Profil bewertet und auf diese Weise eine Rangliste erstellt.

Impliziter Fall: Diese Ausprägung der ähnlichkeitsbasierten Gruppenbildung ist der nutzenbasierten Gruppenbildung ähnlich.

Es ist für diesen Fall - je nach Algorithmus - eine spezielle Payoff-Funktion $\tilde{\pi}$ definiert. Zusätzlich wird ein Parameter k benötigt, der spezifiziert, in wie viele Gruppen die Teilnehmer aufgeteilt werden sollen. Dieser Vorgabewert kann - sofern er nicht direkt spezifiziert wird - aus der Gruppengröße und aus der Gruppendefinition abgeleitet werden.

Um festzustellen, ob ein Profil einem anderen gegenüber *ähnlich* ist, muss - wie bereits in Definition 4.3.4.1 eine Distanzfunktion d auf das Profil definiert sein (siehe Abschnitt 3.3.1.1).

Nachfolgend werden zwei Verfahren zur Lösung dieses Problems definiert. Begonnen wird mit einer Variante des k-Means Algorithmus, der bereits in Abschnitt 3.3.1.2 thematisiert wurde. Dort wurde erwähnt, dass die Kostenfunktion in Gleichung 3.17 zu minimieren ist. Diese Kostenfunktion $\tilde{\pi}$ wird somit zu

$$\tilde{\pi}(\mathcal{G}_i) = \sum_r \sum_s d(x_r, y_s)^2. \tag{4.12}$$

Um eine Aufteilung in k Gruppen zu erreichen muss gelten:

$$\min \sum_{i=1}^{k} \tilde{\pi}(G_i), \tag{4.13}$$

d. h. die Payoff-Funktion $\tilde{\pi}$ muss minimal werden. Die Suche nach einem globalen Minimum ist jedoch sehr aufwändig, weshalb im Folgenden ein iteratives Verfahren vorgestellt wird, welches ein lokales Minimum erreicht. Der Algorithmus zielt darauf ab, ausgehend von einer initialen Gruppierung diese sukzessive zu verbessern, bis ein Minimum erreicht ist.

Für den Ablauf ist der Begriff des Centroids notwendig. Sei $\mathbf{p} = (x_1^p, \ldots, x_d^p)$ ein Punkt im d-dimensionalen Profilraum Π. Ein Schwerpunkt bzw. Centroid μ_C einer Gruppe $\mathcal{G}$ sei definiert durch

$$\mu_C = \left(\bar{x}_1^p, \ldots, \bar{x}_d^p\right) \quad \text{mit} \quad \bar{x}_j^p = \frac{1}{|\mathcal{G}|} \sum_{p \in \mathcal{G}} x_j^p. \tag{4.14}$$

Der Algorithmus läuft in folgenden Schritten ab:

1. Erzeugen von k initialen Gruppen: Die Auswahl erfolgt entweder zufallsbasiert oder systematisch.

2. Zuordnung aller Objekte zu einem der k Gruppenzentren: Jedes Objekt O wird dem Gruppenzentrum μ_C zugeordnet, zu dem der geringste Abstand (spezielle Payoff-Funktion $\tilde{\pi}$) besteht.

3. Neuberechnung der Centroide gemäß Gleichung 4.14

4. Wiederholung der Schritte 2-3 bis keine Neuordnung der Objekte mehr stattfindet (oder Abbruch bei vorgegebenem maximalen Iterationsschritt).

□

Erfolgt die Erzeugung der initialen Gruppen zufällig, so wird die Grundmenge von Objekten willkürlich in k Mengen aufgeteilt. Hartigan [Har75] zeigt, dass diese Vorgehensweise nicht adäquat ist. Er schlägt vor, zuerst den Centroid der kompletten Grundmenge zu berechnen. Die Grundmenge wird in zwei Teilmengen geteilt, indem in einer Dimension sämtliche Punkte, die größer als die zugehörige Schwerpunktkoordinate sind zu einem Teil und der Rest zu einem anderen Teil zusammengefasst wird. Die größere Teilmenge wird abermals nach demselben Schema unterteilt. Dieses Verfahren wird fortgeführt, bis k Teilmengen vorhanden sind.

Danach werden abwechselnd die Gruppierungsobjekte den Centroiden zugewiesen und anschließend die neuen Centroide berechnet.

Da dieses Verfahren in einzelnen Schritten abläuft und das Ergebnis quasi sukzessive verbessert wird, können auch Zwischenergebnisse verwendet werden. Der Algorithmus kann somit den Mobilitätseigenschaften angepasst werden. Es kann eine maximale Iterationsanzahl vorgegeben werden und es muss nicht bis zum Terminieren des Verfahrens verharrt werden.

Für den iterativen Ansatz des k-Means Algorithmus beläuft sich die rechnerische Komplexität bei n zu gruppierenden Objekten in k Gruppen und t benötigten Iterationen auf $\mathcal{O}(tnk)$, wobei i. A. $k, t \ll n$, da die Konvergenz langsam erfolgt. Der iterative k-Means Algorithmus liefert i. A. kein globales Optimum und das Ergebnis der Aufteilung ist von der Reihenfolge abhängig, was sich in Abschnitt 4.3.4.3 noch als negativ erweisen wird.

Der zweite Algorithmus für die implizite Gruppierung ist *Partitioning Around Medoids* (PAM), welcher von Kaufman und Rousseeuw [KR90] stammt.

Im Gegensatz zu k-Means Algorithmus dient bei PAM ein reales Gruppierungsobjekt als Mittelpunkt. Diese werden als *Medoide* bezeichnet. Von allen Gruppierungsobjekten O_i einer Gruppe wird ein Objekt zu einem Medoid O_M, wenn es dem Centroid μ_C am nächsten liegt. Ferner müssen noch Maße für eine Gruppe ($\tilde{\pi}$) und für die komplette Gruppenbildung existieren. Als Payoff-Funktion $\tilde{\pi}$ einer Gruppe dient die Summe der Abstände zum Medoid

$$\tilde{\pi}(\mathcal{G}) = \sum_{O_i \in \mathcal{G}} \tilde{\pi}(\mathcal{G}) = \sum_{O_i in} d(O_i, O_M). \tag{4.15}$$

Gleichung 4.15 unterscheidet sich von Gleichung 4.12 dadurch, dass nicht die quadrierten Abstände aufsummiert werden, sondern nur die einfache Distanz. Grund hierfür ist, dass

durch das Quadrieren große Distanzen überproportional in die Gleichung mit eingehen und deshalb der k-Means Algorithmus sehr sensibel auf Ausreißer reagiert. Deshalb wird bei PAM auf das Quadrieren verzichtet.

Die Aufteilung in k Gruppen erfolgt wie bereits in Gleichung 4.13 angegeben durch

$$\min \sum_{i=1}^{k} \tilde{\pi}(G_i). \tag{4.16}$$

PAM benötigt ferner eine Distanzmatrix $\mathbf{D}_{ij}$ (siehe Gleichung 3.16). Das ist notwendig, da des Öfteren die Abstände der einzelnen Punkte untereinander benötigt werden und sie durch die Distanzmatrix gegeben sind, so dass die einzelnen Distanzen nur einmalig berechnet werden müssen.

Der Algorithmus ist bei Ester und Sander [ES00] oder Ng und Han u. a. [NH02] angegeben und ein Schritt läuft folgendermaßen ab:

1. Auswahl von k repräsentativen Objekten als Medoide.
2. Berechne für jedes Paar (Medoid O_M, Nicht-Medoid O_N) die Werte $\tilde{\pi}(\mathcal{G}_i)$ unter der Annahme, dass O_M durch O_N ersetzt wird.
3. Ist $\sum_{i=1}^{k} \tilde{\pi}(G_i)$ für den vertauschten Fall kleiner als im nicht vertauschten Fall, ersetze O_M durch O_N und zurück zu Schritt 2.
4. Im anderen Fall wird jedem O_N der nächst gelegene O_M zugewiesen.

□

Hinsichtlich der Wahl des initialen Zustandes kann wie beim k-Means Verfahren vorgegangen werden. In der Algorithmusbeschreibung bei Ester und Sander [ES00] wird als erster Medoid dasjenige Objekt O_i herangezogen, welches $\tilde{\pi}(O_i)$ minimiert. Die nächsten Medoide werden für die Objekte berechnet, die nicht das erste Medoid als nächst gelegenes Objekt haben. So wird für alle weiteren benötigten Medoide verfahren. In Ng und Han [NH02] wird anstelle dieses doch komplexen Verfahrens ein zufallsbasierter Ansatz vorgeschlagen, bei dem willkürlich k Objekte als Medoide festgelegt werden.

In Schritt zwei muss zusätzlich hinzugefügt werden, dass mehrere Vertauschungen von O_M und O_N zu kleineren Werten von $\sum_{i=1}^{k} \tilde{\pi}(G_i)$ führen können. In diesem Fall werden die O_M und O_N vertauscht, für die die Summe am niedrigsten ist.

Obige Beschreibung des PAM-Algorithmus spiegelt nur einen Durchgang wider. Um ein globales Optimum zu finden, muss er mehrmals durchgeführt werden, bis keine Verbesserung mehr auftritt. Der Aufwand pro Iteration des PAM Algorithmus beläuft sich auf $\mathcal{O}(k(n-k)^2)$, wenn n Objekte in k Gruppen einteilbar sind. Der primäre Vorteil des Algorithmus ist, dass das erhaltene Ergebnis nicht von der Reihenfolge der Gruppierungsobjekte abhängig ist.

Für den PAM Algorithmus existieren mehrere Erweiterungen. Auf Grund des großen Rechenaufwandes eignet er sich nur für kleine Gruppierungsanwendungen. Deshalb stellen Kaufman und Rousseeuw [KR90] eine Erweiterung für größere Datenmengen (∼1000 zu gruppierende Objekte) dar. Dadurch dass jedoch die Anzahl an Rechenschritten reduziert wird, geht die Optimalitätseigenschaft von PAM verloren.

Von der Laan *et al.* [vdLPB02] haben festgestellt, dass sich der PAM Algorithmus nicht optimal verhält, wenn neben großen Gruppen auch kleine vorhanden sind. Deshalb schlagen sie eine Änderung des PAM dahingehend vor, dass sie eine andere Maximierungsbedingung definieren. Da diese jedoch nicht relevant für MoPiDiG ist, sei an dieser Stelle auf das Papier [vdLPB02] der Autoren verwiesen.

4.3.4.3 Globale Gruppenbildung

Nachdem jeder Knoten im Segment eine lokale Gruppe erzeugt hat, müssen aus allen lokalen Gruppen eine oder mehrere globale Gruppen erzeugt werden. Dieser Vorgang besteht aus der Verteilung der lokalen Gruppen an diejenigen Knoten im Segment, die bisher noch nicht Teil der lokalen Gruppe sind. Wird eine lokale Gruppe von einem Knoten empfangen, so muss diese mit der eigenen lokalen Gruppe kombiniert werden, d. h. es muss überprüft werden, wie die zusätzlichen Informationen zu einer globalen Gruppe vereinigt werden können.

Formale Definition: Bevor die algorithmischen Einzelheiten der Kombination von Gruppen näher erläutert wird, wird ein Kombinationsoperator γ eingeführt. Anhand dieses Operators sollen die notwendigen Eigenschaften und Forderungen der Kombination von Gruppen verdeutlicht werden.

Definition 16: *Kombinationsoperator* γ
Seien $\mathcal{G}_i$ und $\mathcal{G}_j$ zwei lokale Gruppen und v ein willkürlicher Knoten aus $\mathcal{G}_i \cup \mathcal{G}_j$, so ist $\gamma : \mathfrak{P}(V) \times \mathfrak{P}(V) \rightarrow \mathfrak{P}(V)$ eine partielle Vereinigung von $\mathcal{G}_i$ und $\mathcal{G}_j$ mit $\gamma(\mathcal{G}_i, \mathcal{G}_j) = \mathcal{G}_i \cup \mathcal{G}_j \setminus A$, wobei gelten muss:

$$\big|\gamma(\mathcal{G}_i, \mathcal{G}_j)\big| \geq |\mathcal{G}_i| \quad \vee \quad \big|\gamma(\mathcal{G}_i, \mathcal{G}_j)\big| \geq |\mathcal{G}_j| \tag{4.17}$$

$$\epsilon\big(v, \gamma(\mathcal{G}_i, \mathcal{G}_j)\big) = \texttt{true} \tag{4.18}$$

$$\pi\big(\gamma(\mathcal{G}_i, \mathcal{G}_j)\big) \geq \pi(\mathcal{G}_i) \quad \wedge \quad \pi\big(\gamma(\mathcal{G}_i, \mathcal{G}_j)\big) \geq \pi(\mathcal{G}_j) \tag{4.19}$$

$$\pi\big(\gamma(\mathcal{G}_i, \mathcal{G}_j)\big) = \max_{\forall M \in \mathfrak{P}(\mathcal{G}_i \cup \mathcal{G}_j)} \pi(M) \tag{4.20}$$

Die erste Forderung stellt sicher, dass durch die Kombination zweier Gruppen die Gruppengröße nicht abnimmt. Der zweite Punkt garantiert, dass die Kombination zweier Gruppen ebenfalls eine Gruppe gemäß des ϵ-Operators darstellt. Gleichung 4.19 garantiert, dass mit der Kombination der Gruppen ein *Nutzen* verbunden ist. Die letzte Forderung macht die Kombination eindeutig, indem der maximale Nutzen gefordert wird.

Hinsichtlich der Eigenschaften[14] von γ wären folgende als positiv zu bewerten:

$$\text{Kommutativ: } \gamma(\mathcal{G}_i, \mathcal{G}_j) = \gamma(\mathcal{G}_j, \mathcal{G}_i), \tag{4.21}$$

$$\text{Assoziativ: } \gamma(\gamma(\mathcal{G}_i, \mathcal{G}_j), \mathcal{G}_k) = \gamma(\mathcal{G}_i, \gamma(\mathcal{G}_j, \mathcal{G}_k)). \tag{4.22}$$

Diese Forderungen würden garantieren, dass die Kombination unabhängig von der Reihenfolge der einzelnen Gruppen ist und somit die schließlich erhaltene globale Gruppe für jeden Knoten identisch ist. Allerdings sind dies sehr restriktive Anforderungen an γ und

[14] Notationshinweis: Im Folgenden sei $\gamma(\ldots(\gamma(\mathcal{G}_1, \mathcal{G}_2), \ldots), \mathcal{G}_n)$ mit $\gamma(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n)$ abgekürzt. Soll γ auf eine komplette Menge M angewandt werden, soll dies mit $\gamma(M)$ notiert sein.

können nicht als gegeben angenommen werden. Für das MoPiDiG Framework bedeutet dies, dass eine Lösung dargelegt werden muss, die die Kommutativität und Assoziativität nicht fordert.

Mit dieser Definition des Kombinationsoperators γ kann jetzt die Definition einer Globalen Gruppe angegeben werden.

Definition 17: *Globale Gruppe C*
Sei N_v^k die Nachbarschaftsmenge eines Knoten v und $\mathfrak{G}$ die Menge der lokalen Gruppen aller Knoten $v_j \in N_v^k$. Ferner sei $\mathfrak{P}(\mathfrak{G})$ die Potenzmenge von den lokalen Gruppen $\mathfrak{G}$ und $M \in \mathfrak{P}(\mathfrak{G})$, dann muss für eine Globale Gruppe C gelten:

$$v_i \in C : \quad v_i \in \gamma(M)$$

□

Die Globale Gruppe C besteht somit aus allen Knoten aus N_v^k, die gemäß ϵ und γ *ähnlich* sind.

Nachdem eine Globale Gruppe formal definiert und somit bekannt ist, dass für den Erhalt einer Globalen Gruppe der Austausch von lokalen Gruppen durchgeführt werden muss, wird das Verteilen der lokalen Gruppen im nächsten Abschnitt erläutert. Danach wird der Kombiniationprozess von lokalen Gruppen und damit die Verbindung zum Kombinationsoperator *gamma* näher beschrieben.

Verteilen der Lokalen Gruppen: Der lokale Gruppierungsvorgang liefert für jeden Knoten eine Vorauswahl, mit welchen direkten Nachbarn er eine Gruppe bilden will. Um eine im Netzwerk oder Segment optimale Gruppe zu erreichen, müssen die lokalen Sichten ausgetauscht werden. Hierfür ist ein Protokoll nötig, welches festlegt, wie die einzelnen lokalen Sichten auszutauschen sind.

Eine Einigung könnte z. B. erreicht werden, indem sämtliche Knoten ihre Gruppenvorschläge an alle anderen Knoten senden und jeder Knoten für sich intern abstimmt, mit welchen anderen Knoten eine Gruppe gebildet wird. Bei dieser Methode werden jedoch viele Nachrichten versendet und die Probleme, die durch die Mobilität der Teilnehmer auftreten werden nicht berücksichtigt.

Es existieren aber bereits andere Verfahren, die als Einigungsalgorithmen in der Literatur bekannt sind. Singhal und Shivaratri [SS94] geben einen Überblick über verschiedene Verfahren.

Einigungsalgorithmen: Bei den Einigungsalgorithmen einigen sich n Teilnehmer auf einen gemeinsamen Wert. In den Annahmen von Einigungsalgorithmen ist ferner enthalten, dass eine Teilmenge der am Einigungsprozess beteiligten Knoten bewusst fehlerhaft sein darf. Wie groß diese Teilmenge sein darf, hängt von den weiteren Annahmen ab.

Da bei Einigungsalgorithmen die Art der Kommunikation wesentlich ist, wird kurz der Unterschied zwischen synchroner und asynchroner Kommunikation beschrieben, bevor existierende Verfahren vorgestellt und hinsichtlich ihrer Einsetzbarkeit bewertet werden.

Asynchrone und Synchrone Kommunikation: Bei der asynchronen Kommunikation bestehen bezüglich des Sendens einer Nachricht aus zeitlicher Sicht keinerlei Einschränkungen. Nachrichten können zu jedem Zeitpunkt gesendet und empfangen werden. Der Sender blockiert nicht nach Versenden einer Nachricht.

Die synchrone Kommunikation läuft in einzelnen Runden ab. Wenn ein Sender eine Nachricht verschickt, so führt er keine Aktionen (d. h. er blockiert) aus, bis er eine Antwort erhält. Jedem Teilnehmer ist bekannt, welche und wie viele Nachrichten in welcher Runde eintreffen müssen und wie er zu antworten hat. Langsamer Nachrichtenaustausch verzögert demgemäß das komplette System. Um unendliche Wartezeiten - wie sie bei einem Rechnerausfall durchaus in Erscheinung treten können - zu verhindern sind Zeitüberschreitungsmechanismen (z. B. Timeouts) einzusetzen.

Existierende Einigungsalgorithmen: Der primäre Vorteil von Einigungsalgorithmen ist das Erzielen einer Einigung, obwohl manche Teilnehmer fehlerhaft arbeiten. Für den Gruppierungsprozess bedeutet dies, dass unter den zu gruppierenden Teilnehmern eine Teilmenge existiert, die absichtlich falsche Informationen weiterleitet. Die Obergrenze [PSL80] an fehlerhaften Knoten ist $\lfloor \frac{n-1}{3} \rfloor$, wobei dieser Wert nicht von allen Algorithmen erreicht wird.

Fischer *et al.* [FLP85] beschreiben, dass das Einigungsproblem bei asynchroner Kommunikation mit einem fehlerhaften Knoten nicht lösbar ist, d. h. kein total korrekter Algorithmus existiert, sondern nur ein partiell korrekter[15].

De Prisco *et al.* [PMR99] definieren ein Unterproblem der Einigungsprotokolle. Es müssen sich nicht alle Prozessoren auf einen Wert, sondern auf k-Werte festlegen. Dies hat zur Folge, dass mehr Teilnehmer fehlerhaft sein dürfen als oben erwähnt. Für die Gruppierung bedeutet das, dass als Ergebnis nicht nur eine Gruppe, sondern k Gruppen als Resultat vorliegen könnten.

Dolev *et al.* [DLP$^+$86] konzentrieren sich auf *Approximate Agreement*. Die Einigung erfolgt nicht auf einen festen Wert, sondern liegt innerhalb eines Intervalls. Dieses Problem lässt sich dann auch asynchron lösen (vgl. dagegen Fischer *et al.* [FLP85]). Die Anzahl an korrekten Teilnehmern muss im synchronen Fall mindestens drei mal größer sein, im asynchronen Fall fünf mal größer als die Anzahl der defekten Teilnehmer.

Einsetzbarkeit: Hinsichtlich der Einsetzbarkeit von Einigungsalgorithmen muss gesagt werden, dass sie in ihrer existierenden Form nur bedingt geeignet sind das vorliegende Problem der Gruppenkombination zu lösen.

Bei Einigungsalgorithmen besitzt jeder teilnehmende Knoten i ein bestimmtes Merkmal M_i. Das Ergebnis eines Einigungsalgorithmus ist ein bestimmtes Merkmal M_i, nicht jedoch wie bei MoPiDiG eine Mischform mehrerer M_i.

Verwendeter Algorithmus: Da sich die soeben beschriebenen Einigungsalgorithmen nur bedingt zur Lösung eignen, werden im Folgenden die Protokolle zum Austausch von Profilinformationen beschrieben, damit nach Terminierung des Verfahrens jeder Knoten im Segment über die gleiche Information verfügt.

[15] Ein partiell korrekter Algorithmus erfüllt zwar die Problemspezifikation, muss aber nicht terminieren.

Es werden mehrere Verfahren beschrieben. Die Protokolle unterscheiden sich je nach verwendeter virtueller Topologie.

Verteilung der Information ohne virtuelle Topologie: Liegt keine virtuelle Topologie vor, erfolgt die Verteilung der Gruppeninformation durch direkte Weiterleitung der Daten, bis jeder Knoten sämtliche lokalen Gruppen eines jeden anderen Knotens besitzt.

Abbildung 4.32 informiert in einer Sequenz von Bildern, wie der Verteilungsprozess

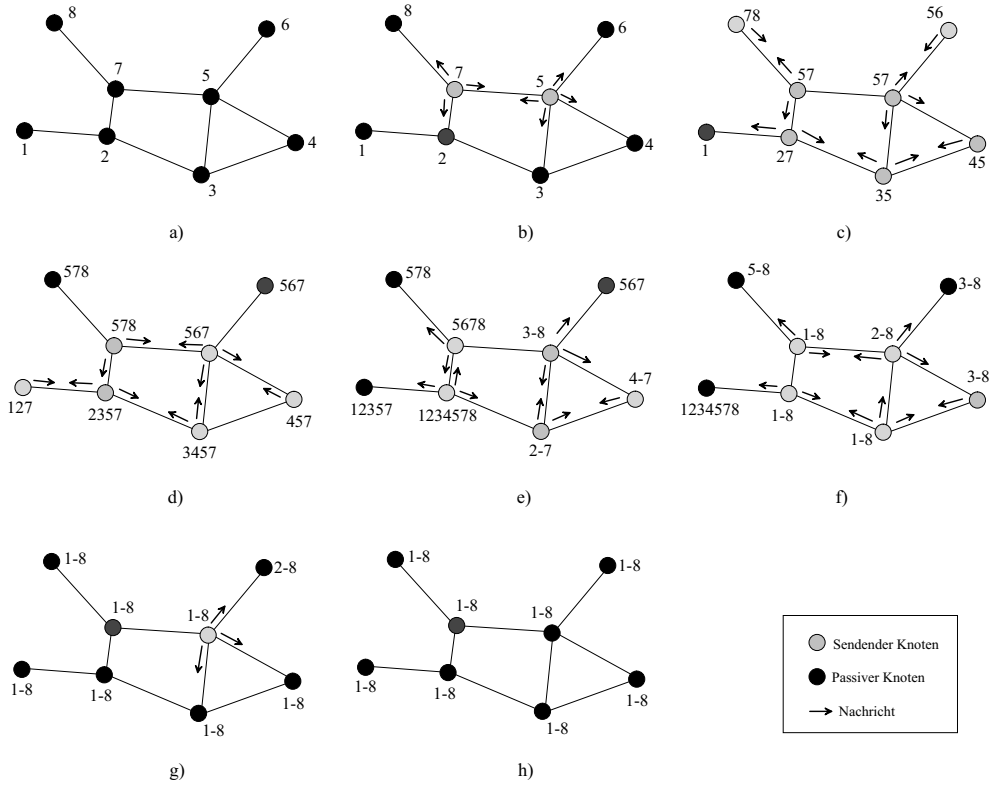


Abbildung 4.32: Verteilung der lokalen Gruppen mit Hilfe eines modifizierten Echo Algorithmus

abläuft. Abbildung 4.32a zeigt das Segment mit Knoten und Identifikatoren.

Initiiert wird der Vorgang von allen Nicht-Randknoten. Dies sind in Abbildung 4.32c die Knoten 5 und 7. Diese Knoten senden ihre lokale Gruppe an alle ihre direkten Nachbarn. Wird eine Nachricht empfangen, so muss zuerst überprüft werden, ob in dieser ein Mehrwert, d. h. neue lokale Gruppen, enthalten sind. Ist dies der Fall, so wird diese Nachricht wieder an alle Nachbarn weitergeleitet. Ist in einer Nachricht keine neue Information enthalten, erfolgt auch keine Weiterleitung. Dieses Verhalten garantiert gleichzeitig die Terminierung des Verfahrens. Haben alle Knoten die gleichen Informationen, werden keine Nachrichten mehr gesendet und jeder Knoten verfügt über sämtliche Profilinformationen, wie schließlich in Abbildung 4.32h zu sehen ist.

Hinsichtlich der Anzahl an versandten Nachrichten ist dieses Verfahren nicht optimal.

So senden sich in Abbildung 4.32f die Knoten 3 und 5 die gleichen Daten zu, da beide von unterschiedlichen Elternknoten eine neue Information erhalten haben. Mit einer Topologie kann dies verhindert werden, wie im Folgenden gezeigt wird.

Verteilung der Information mit vorhandener Baumstruktur: Wird ein Segment mit einer Baumtopologie überlagert, ist die Verteilung der Informationen effizienter gestaltbar.

Unter Verwendung des Echo Algorithmus [Cha82; Seg83] wären zwei Baumdurchläufe nötig. Im ersten Durchlauf werden die lokalen Gruppen gesammelt, so dass am Ende mindestens ein Knoten (der Initiatorknoten) existiert, der sämtliche Informationen hat. Im zweiten Durchlauf würden die lokalen Gruppen so verteilt werden, dass anschließend sämtliche Knoten alle lokalen Gruppeninformationen besitzen. In der Summe wären $3 \cdot (n-1)$ Nachrichten notwendig, damit jeder Knoten alle Profilinformationen bekommt.

Da die Elternknoten der Blattknoten durch die lokale Gruppierung auch über die Profilinformationen der Blattknoten verfügen, kann im ersten Durchlauf auf diese Nachrichten verzichtet werden. In der zweiten Runde können sie nicht eingespart werden, da auch die Blattknoten die komplette Profilinformation benötigen. Somit werden so viele Nachrichten eingespart, wie der Baum Blattknoten besitzt.

In Abbildung 4.33 wird das Segment, welches bereits aus Abbildung 4.32 bekannt ist,

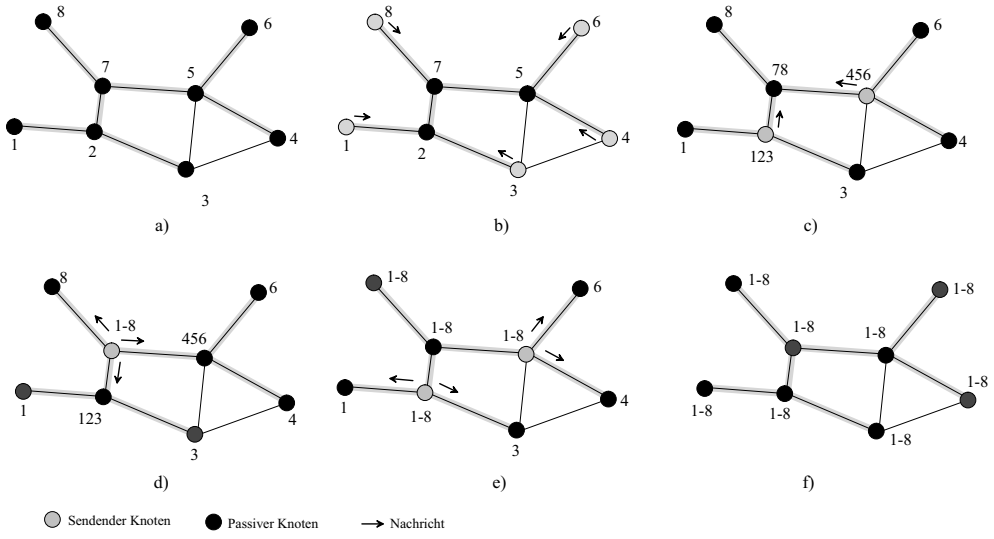


Abbildung 4.33: Verteilung der lokalen Gruppeninformation bei vorhandener Baumstruktur

mit einer Baumtopologie überlagert. Initiiert wird die Verteilung in Abbildung 4.33c durch die Blattknoten. Bekannt sind diese dadurch, dass sie nur eine ausgehende Baumkante besitzen. Diese Knoten senden ihre Profilinformation an ihre Elternknoten, bis schließlich alle Informationen am Wurzelknoten ankommen (Abbildung 4.33d) und von diesem wieder in Richtung Blattknoten verteilt werden (Abbildung 4.33e). In Abbildung 4.33f ist die Verteilung abgeschlossen. Eine Terminierungsbedingung ist nicht notwendig, da die Terminierung über die Topologie geregelt wird. Ein Elternknoten wartet mit der Weiterleitung

bis sämtliche Kinderknoten ihre lokalen Gruppen versendet haben.

Bei dem Verfahren bekommen die Blatt- bzw. Randknoten die vollständige Gruppeninformation definitiv zuletzt. Beim Verfahren ohne Topologie ist die Reihenfolge zufällig. Prinzipiell sollten die Randknoten zuerst die komplette Gruppeninformation erhalten, da für diese Knoten die Tendenz am größten ist, dass sie das Segment verlassen. Inwiefern sich diese Tatsache negativ auswirkt, kann jedoch nicht allgemein beurteilt werden und bedürfte weiterer eingehender Untersuchungen.

Aus dieser Beschreibung ergibt sich auch, dass - sofern eine Baumtopologie vorhanden ist - die Blattknoten keine lokale Gruppe erzeugen müssen, da die zusätzlich gewonnenen Informationen nicht verwertet werden.

Verteilung der Information mit vorhandener Ringtopologie: Die Verteilung mit Hilfe eines Ringes ist quasi offensichtlich. Werden Informationen einmal durch den Ring gesandt, so hat sie jeder Teilnehmer im Ring. Mit maximal zwei Ringdurchläufen können alle Nachrichten propagiert werden, wie Abbildung 4.34 zeigt.

In Abbildung 4.34 ist ein Segment mit einer logischen Ringstruktur überlagert. Es

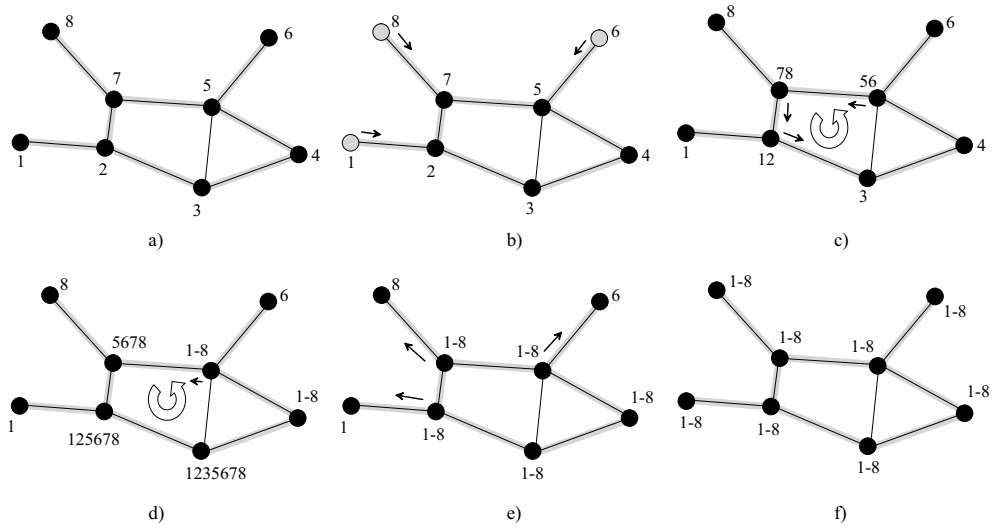


Abbildung 4.34: Verteilen der lokalen Gruppen bei einer Ringstruktur

handelt sich um einen logischen Ring, da noch zusätzliche „Seitenarme“ vorhanden sind. In Abbildung 4.34b starten die Knoten den Verteilungsprozess, die sich am Ende der Seitenarme befinden. Diese senden ihre lokalen Gruppeninformationen an den Nachbarn, der wiederum seine lokale Gruppe hinzufügt und in den Ring weiterleitet. Abbildung 4.34d zeigt die Situation nach dem ersten Durchlauf. Der Initiator besitzt bereits sämtliche Profilinformationen und muss diese durch einen erneuten Durchlauf bekannt machen. Sobald ein Knoten die zweite Nachricht erhalten hat, können auch sämtliche „Seitenarme“ des Rings informiert werden. Somit erfolgen die Vorgänge in den Abbildungen 4.34d und 4.34e nicht sequentiell sondern parallel. Abbildung 4.34f zeigt, wie nach zwei Durchgängen jeder Knoten über alle Informationen verfügt.

Kombination Lokaler Gruppen: Hinsichtlich der Bildung globaler Gruppen muss zwischen den folgenden Gruppierungsarten unterschieden werden.

Nutzenbasiertes Gruppieren: Um eine lokale Gruppe zu bilden, wurde aus den Profilen der direkten Nachbarn die Teilmenge gewählt, die den Nutzen eines Teilnehmers maximiert. Als Modell diente das Profilnetzwerk, welches sämtliche Profilpunkte als einen vollvermaschten gewichteten Graphen darstellt. Eine lokale Gruppe eines Punktes P ist schließlich der kürzeste Pfad im Profilnetzwerk, der das Profil von P enthält.

Um eine globale Gruppe zu erhalten, muss in dem Profilnetzwerk, welches aus *allen Profilen* der Segmentmitglieder besteht, die Teilmenge gefunden werden, welche den größten Nutzen ergibt.

Hinsichtlich der Lösung dieses Problems, existieren All-Pair-Shortest Path Algorithmen für gewichtete Graphen, wie z. B. der Floyd-Warshall-Algorithmus [CLRS01]. Diese Verfahren nutzen jedoch nicht die bereits vorhandenen lokalen Gruppen und können auch nur verwendet werden, wenn die Gewichte im Graphen statisch sind.

Die Kombination der einzelnen lokalen Gruppen hat so zu erfolgen, dass am Ende sämtliche Knoten im Segment über die gleiche Gruppenzusammensetzung verfügen. Die Kommutativität ist als erfüllt zu betrachten, da jeweils die Gruppe mit dem für ihre Mitglieder größeren Nutzen Ausgangspunkt für eine Kombination ist.

Ein assoziatives Verfahren würde dies gewährleisten, ist aber nicht zwingend erforderlich. Es reicht aus, wenn das Endergebnis nach der Bildung aller Kombinationen gleich ist. Bei vier lokalen Gruppen bedeutet dies, dass zwar für eine Kombination von drei Gruppen als Zwischenergebnis

$$(A \circ B) \circ C \neq A \circ (B \circ C) \quad \text{und} \tag{4.23}$$
$$(B \circ C) \circ D \neq B \circ (C \circ B) \tag{4.24}$$

tolerierbar ist, aber schließlich

$$(A \circ B) \circ (C \circ D) = A \circ (B \circ C) \circ D = A \circ B \circ C \circ D \tag{4.25}$$
$$A \circ B \circ C \circ D = D \circ A \circ B \circ C = C \circ D \circ A \circ B = ... \tag{4.26}$$

erfüllt sein muss. Letztendlich muss jede mögliche Permutation von A, B, C und D das gleiche Ergebnis liefern wie $A \circ B \circ C \circ D$.

Bereits bei der lokalen Gruppierung wurde festgestellt, dass der Test sämtlicher Permutationen zu viel Zeit in Anspruch nimmt. Aus diesem Grund muss ein effizient-systematisches Verfahren angewandt werden. Ausgangspunkt für jeden Punkt ist hierfür jeweils die bereits bestimmte lokale Gruppe $\mathcal{G}_i$. Wird von einem Segmentnachbarn eine andere lokale Gruppe $\mathcal{G}_j$ empfangen, gibt es folgende Möglichkeiten:

$$\mathcal{G}_i \subseteq \mathcal{G}_j \quad (i \neq j) \tag{4.27}$$
$$\mathcal{G}_i \cap \mathcal{G}_j = \emptyset \quad (i \neq j) \tag{4.28}$$
$$\mathcal{G}_i \cap \mathcal{G}_j \neq \emptyset \quad (i \neq j) \tag{4.29}$$

Tritt der Fall gemäß Gleichung 4.27 ein, ist keine Kombination der Gruppierung notwendig, da eine Gruppe eine Teilmenge der anderen ist.

Im zweiten Fall müssen sämtliche neu hinzukommenden Gruppierungsobjekte so in die bisherige Gruppe eingefügt werden, dass sich π erhöht.

Gleichung 4.29 ist der „Normalfall“, da zwei benachbarte lokale Gruppen i. A.[16] gleiche Gruppierungsobjekte enthalten. Dieser Fall kann, wenn die Schnittmenge beider Gruppen subtrahiert wird, auf den zweiten Fall reduziert werden. Denn für die Kombination zweier Gruppen nach dem dritten Fall gilt

$$\pi(\mathcal{G}_i \cup \mathcal{G}_j) = \pi\Big(\mathcal{G}_i \cup \big(\mathcal{G}_j \backslash (\mathcal{G}_i \cap \mathcal{G}_j)\big)\Big).$$

Um entscheiden zu können, ob mit der Kombination ein Mehrwert erreicht wird, muss der Kombinationsoperator γ definiert werden. Hierzu müssen aus der Payoff Funktion π Bedingungen an die Gewichte der zukünftigen Gruppenmitglieder abgeleitet werden, die erfüllt sein müssen, damit sich der Wert von π erhöht. Es werden folgende Bedingungen benötigt:

$$\begin{aligned}
\text{An- bzw. Einfügen eines Objektes } O\text{:} \quad & \pi(\mathcal{G} \cup O) > \pi(\mathcal{G}_i) && (4.30)\\
\text{Löschen eines Objektes } O\text{:} \quad & \pi(\mathcal{G}_i \setminus O) > \pi(\mathcal{G}_i) && (4.31)\\
\text{Ersetzen eines Objektes } O \text{ durch } U\text{:} \quad & \pi\big((\mathcal{G}_i \setminus O) \cup U\big) > \pi(\mathcal{G}_i) && (4.32)\\
\text{Anfügen mehrerer Objekte } O_i\text{:} \quad & \pi(\mathcal{G}_i \cup O_i) > \pi(\mathcal{G}_i) && (4.33)
\end{aligned}$$

Liegt z. B. eine Bedingung gemäß Gleichung 4.31 vor, kann nur durch Betrachtung der Gewichte eruiert werden, ob sich π durch das Löschen eines Elements erhöht.

Mit den Gleichungen 4.30-4.33 kann ein Algorithmus für das globale Gruppieren angegeben werden.

1. Löschen von Gruppenteilnehmern gemäß Gleichung 4.31, bis Löschen keinen Profitgewinn mehr erbringt.

2. Anfügen eines Gruppierungsobjekts gemäß Gleichung 4.30. Anschließend muss erneut überprüft werden, ob das Löschen von Gruppenteilnehmern Profitgewinn erbringt. Diesen Vorgang so lange wiederholen, bis kein Profitgewinn mehr erzielt wird.

3. Test auf Ersetzen gemäß Gleichung 4.32.

4. Nach Gleichung 4.33 nach potenziellen Gruppierungsobjekten suchen.

5. Wurden im vorherigen Schritt Gruppierungsobjekte hinzugefügt, zurück zu Schritt 1, ansonsten Kombination der Lokalen Gruppen beendet.

Ein Durchgang in Schritt 1 ist von der Ordnung $\mathcal{O}(|\mathcal{G}|)$ und kann somit sehr effizient durchgeführt werden. Das Anfügen ist bei einer Segmentgröße von n von der Ordnung $\mathcal{O}(n - |\mathcal{G}|)$. Beim Überprüfen auf ein mögliches Ersetzen müssen die $|\mathcal{G}|$ Gruppenobjekte mit den $n - |\mathcal{G}|$ restlichen Segmentmitgliedern verglichen werden. Dies führt zur Ordnung $\mathcal{O}(n|\mathcal{G}| - |\mathcal{G}|^2)$.

[16] Bei benachbarten Gruppen kann die Schnittmenge auch leer sein, z. B. wenn die lokale Gruppe nur aus dem Gruppeninhaber besteht.

Der aufwändigste Schritt bei diesem Verfahren ist Schritt 4. Hier werden die Fälle behandelt, bei denen das Hinzufügen eines Gruppierungsobjekts keinen Profitgewinn erbringt, jedoch durch die gleichzeitige Aufnahme mehrerer Gruppierungsobjekte ein höherer Nutzen erzielt werden kann. Hier kann jedoch die Anzahl an zu überprüfenden Objekten i. A. durch die Bedingung gemäß Gleichung 4.32 selbst eingeschränkt werden. Gleichung 4.32 gibt Obergrenzen für die Gewichte an. Deshalb können sämtliche Gruppierungsobjekte, die diese Obergrenze überschreiten, ausgeschlossen werden.

Ist die Gruppenerstellung für eine Gruppe beendet und es sind noch Gruppierungsobjekte vorhanden, so kann versucht werden, diese zu weiteren Gruppen zusammenzufasssen. Ausgangspunkt ist die lokale Gruppe, die den größten Nutzen aufweist, nachdem sämtliche Gruppierungsobjekte entfernt wurden, die bereits Mitglied anderer globaler Gruppen sind.

Alternativ zum obigen Verfahren wird noch ein zweites Verfahren vorgestellt, welches auf sehr einfache und schnelle Weise eine globale Gruppe ermittelt.

In den verschiedenen lokalen Gruppen treten die einzelnen Gruppierungsobjekte unterschiedlich häufig auf. In Abbildung 4.35 ist dies ersichtlich. Abbildung 4.35a zeigt ein Segment und in Abbildung 4.35b sind die Profilpunkte der jeweiligen Segmentteilnehmer (zu identifizieren an den Identifikatoren) ersichtlich. Die Abbildungen 4.35c-4.35n bilden jeweils die lokalen Gruppen der Segmentteilnehmer ab.

Die Häufigkeiten der einzelnen Gruppierungsobjekte in den lokalen Gruppen sind in Tabelle 4.3 aufgezeigt. Die Tabelle unterscheidet zwischen absoluten und relativen Häufig-

Gruppierungsobjekt	1	2	3	4	5	6	7	8	9	10
Häufigkeit	I	II	III	-	I	II	III	I	III	II
Anzahl Nachbarn	3	3	3	2	3	4	4	2	3	3
Verhältnis	$\frac{1}{3}$	$\frac{2}{3}$	1	0	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{1}{2}$	1	$\frac{2}{3}$

Tabelle 4.3: Vorkommen der Profilobjekte in den einzelnen lokalen Gruppen

keiten. Die relativen Häufigkeiten ergeben sich aus der absoluten Häufigkeit im Verhältnis zu den tatsächlichen Nachbarn im Segment.

Die Idee ist nun, dass diejenigen Gruppierungsobjekte, die in vielen lokalen Gruppen auftreten, auch in der globalen Gruppe vertreten sein sollen. Diese Objekte können als Ausgangspunkt für obig beschriebenes Verfahren dienen.

Ähnlichkeitsbasiertes Gruppieren: Für den ähnlichkeitsbasierten Ansatz wird wie bereits beim lokalen Gruppieren zwischen dem *expliziten* und *impliziten* Fall unterschieden.

Expliziter Fall: Der explizite Fall ist dadurch gekennzeichnet, dass die Profilattribute direkt angegeben werden, die für eine Gruppenbildung relevant sind. Eine Payoff-Funktion π ist nicht notwendig.

Deshalb ist die Kombination γ von lokalen Gruppen durch den Vereinigungsoperator von Mengen $\gamma = \cup$ gegeben. Da dieser Operator sowohl kommutativ als auch assoziativ ist, ist hinsichtlich der Reihenfolge nichts zu berücksichtigen und kann willkürlich erfolgen.

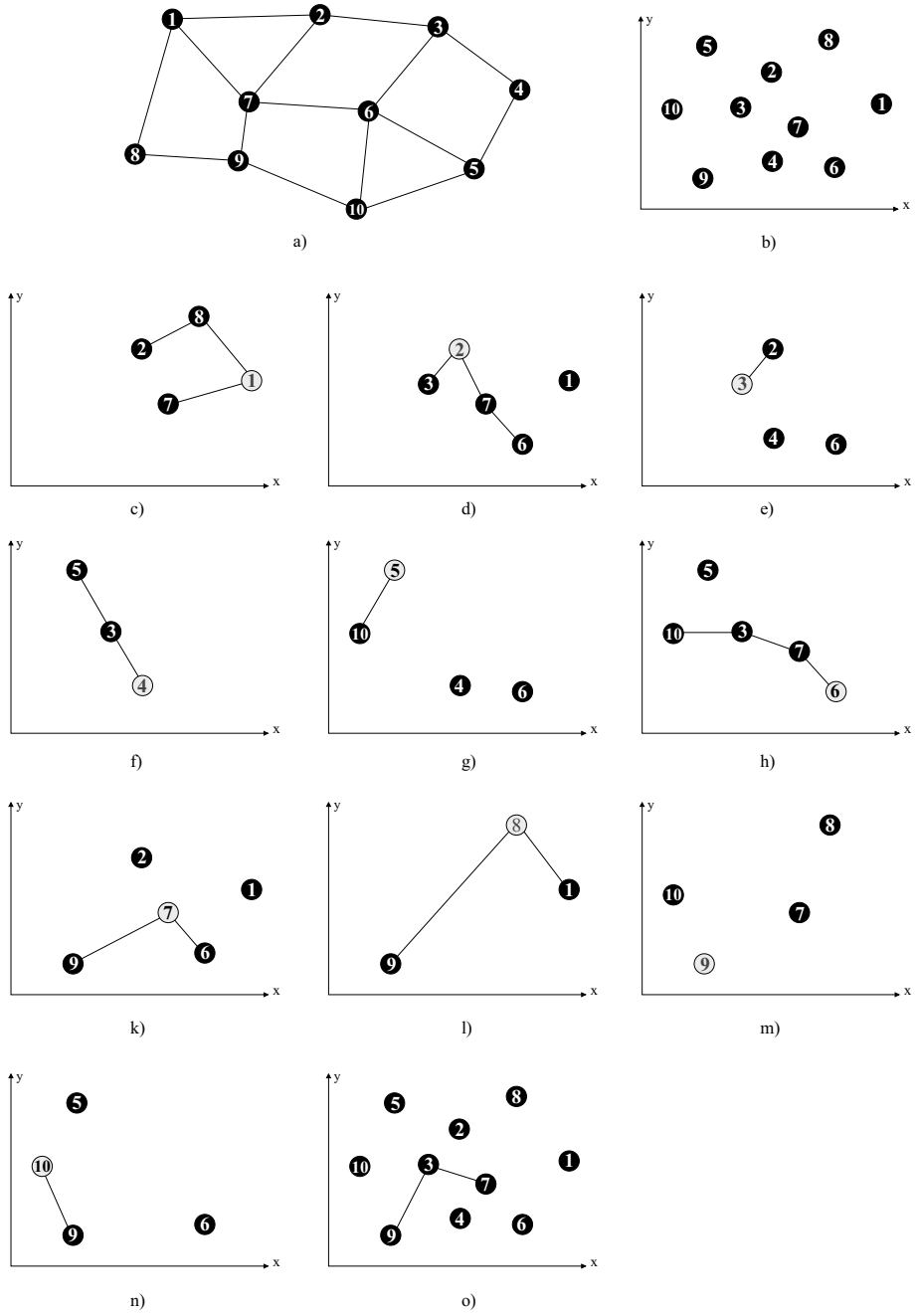


Abbildung 4.35: a) zeigt das Segment und b) sämtliche Profile. c) bis n) zeigen für jedes Segmentmitglied die lokale Gruppe. o) zeigt die initiale globale Gruppe gemäß dem alternativen Verfahren.

Für den Fall, dass eine Konkatenation von Attributen angegeben ist, muss bewertet werden, welcher Knoten die meisten der in der Konkatenation geforderten Attribute

erfüllt. Dies wird erreicht, indem das Ergebnis am Ende sortiert wird. Notwendig ist es allerdings nur, wenn eine maximale Anzahl an Gruppenmitgliedern angegeben wurde und mehr Knoten die Anforderung bzw. einen Teil hiervon erfüllen.

Impliziter Fall: Im impliziten Fall wird eine lokale Gruppe durch eine iterative Variante des k-Means-Algorithmus oder durch den *Partitioning Around Medoids* (PAM) Algorithmus erzielt. Die Kombination besteht - nachdem neue Profildaten erhalten wurden - aus einem Zuordnen der neuen Information zu bestehenden Gruppen mit anschließendem Optimieren durch eine Wiederholung des lokalen Gruppierungsalgorithmus.

Für den k-Means-Algorithmus ist die Kombination von lokalen Gruppen in Abbildung 4.36 veranschaulicht. In Abbildung 4.36a und Abbildung 4.36b sind jeweils die Aus-

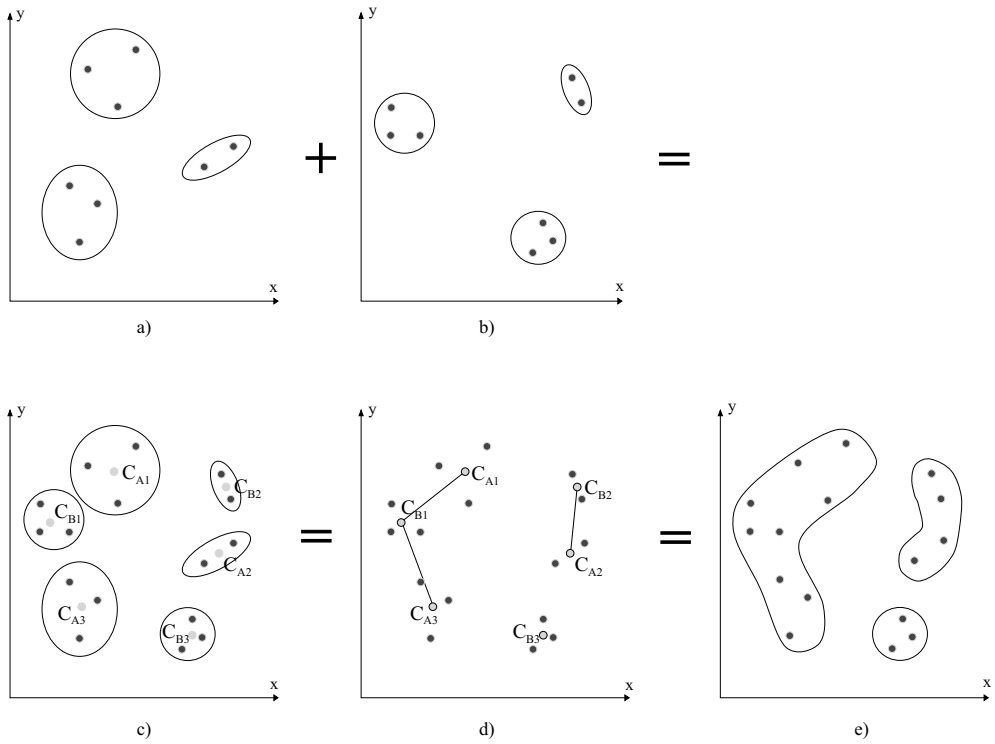


Abbildung 4.36: Verschmelzen zweier lokaler Gruppen mit k=3 beim ähnlichkeitsbasierten Gruppieren

gangsgruppen zu sehen, die miteinander verschmolzen werden sollen. Da k=3 besteht jede lokale Gruppe bereits aus drei Bereichen

Abbildung 4.36c zeigt sämtliche Gruppierungsobjekte in einem Diagramm, wobei zusätzlich noch die Centroide C_i in jeder Gruppe eingezeichnet sind. Der Verschmelzungsvorgang wird eingeleitet, indem die zwei nächsten Gruppen miteinander kombiniert werden und auf diese Weise eine neue Gruppe bilden. Um die Gruppe, die durch den Centroid C_{A1} repräsentiert wird mit einer anderen Gruppe zu kombinieren, wird nach dem nächstgelegenen Centroid gesucht. Dies ist der Centroid C_{B1}. Dem Centroid C_{A2} liegt der Centroid C_{B2}

am nächsten. C_{A3} wird ebenfalls mit C_{B1} kombiniert, obwohl auch schon C_{A1} mit diesem Centroid assoziiert ist. Folglich findet sich für C_{B3} keine Gruppe, da die Anzahl an Gruppen sich ebenfalls nicht ändern darf, soll der k-Wert gleich bleiben. In Abbildung 4.36e ist das Endergebnis dargestellt. Anschließend kann der k-Means Algorithmus durchgeführt werden um das lokale Minimum zu berechnen. Die Tatsache, dass jedoch bereits für die lokalen Gruppen lokale Minima vorlagen, reduziert die Iterationen, die nötig sind, um für die kombinierte Gruppe das Minimum zu erreichen.

Das obig beschriebene Verfahren arbeitet schnell, birgt aber ein erhebliches Problem. Wie Abbildung 4.37 zeigt, ist das Ergebnis nicht kommutativ. Für die Verschmelzung der

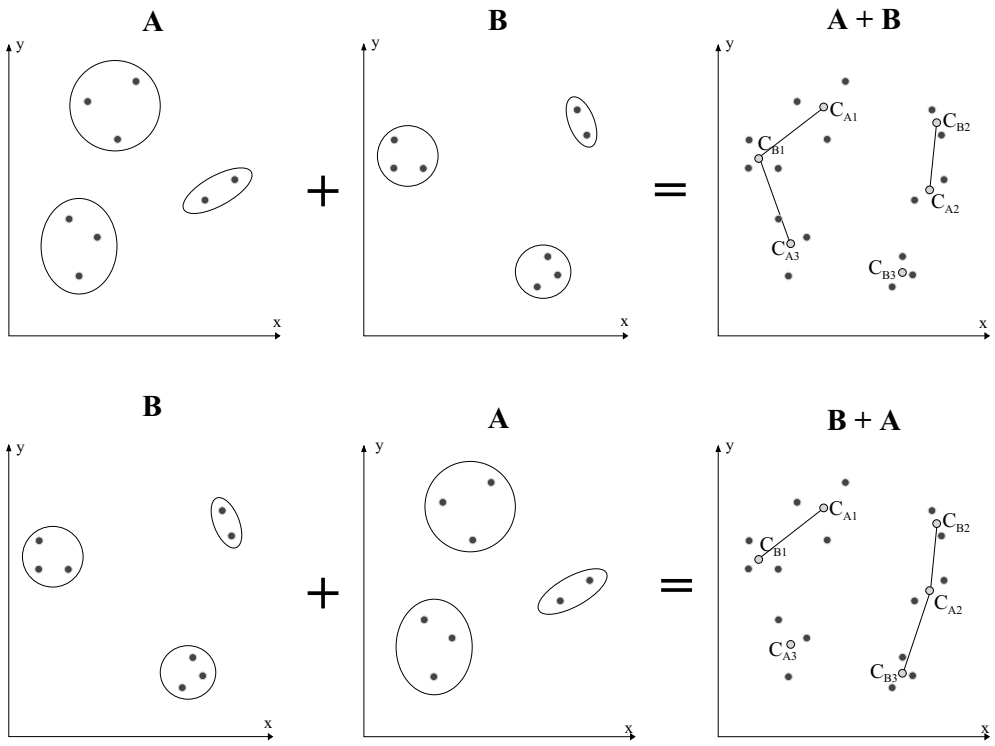


Abbildung 4.37: Die Zuordnung zweier lokaler Gruppen (k=3) anhand der nächsten Centroide ist nicht kommutativ.

Gruppe A mit B wird ein anderes Ergebnis erzielt als für die Verschmelzung der Gruppe B mit A. Auf diese Weise kann folglich keine eindeutige globale Gruppe erzeugt werden. Werden deshalb zwei Gruppen miteinander verbunden, so muss festgelegt werden, welches Ergebnis später weiter verwendet wird. Als Maß hierfür wird die Varianz σ^2 eingeführt.

$$\sigma_i^2 = \frac{1}{|\mathcal{G}_i|} \sum_{i=1}^{|\mathcal{G}_i|} \big(x_i - \mu(\mathcal{G}_i)\big)^2 \tag{4.34}$$

Sind die Varianzen σ_i^2 und σ_j^2 sowie die Mittelwerte $\mu(\mathcal{G}_i)$ und $\mu(\mathcal{G}_j)$ zweier Gruppen $\mathcal{G}_i$, $\mathcal{G}_j$

bekannt, kann die Varianz σ_{ij}^2 der Kombination beider Gruppen durch

$$\sigma_{ij}^2 = \frac{|\mathcal{G}_i|\sigma_m^2 + |\mathcal{G}_j|\sigma_m^2}{|\mathcal{G}_i| + |\mathcal{G}_j|} + \frac{|\mathcal{G}_i| \cdot |\mathcal{G}_j|}{(|\mathcal{G}_i| + |\mathcal{G}_j|)^2}\big(\mu(\mathcal{G}_i) - \mu(\mathcal{G}_j)\big)^2 \tag{4.35}$$

berechnet werden.

Hierdurch liegt ein Verfahren vor, mit dem entschieden werden kann, welches Gruppierungsresultat bei der Kombination zweier Gruppen schließlich Verwendung finden kann. Es werden für alle Gruppen jeweils für die Kombination $A + B$ und $B + A$ die resultierenden Varianzen berechnet. Wie in Abbildung 4.37 ersichtlich, verbinden sich bei einer Kombination $A+B$ die Centroide C_{A_1} mit C_{B_1} und C_{A_3} sowie C_{B_2} mit C_{A_3}. Der Centroid C_{B_3} findet keine Verbindung, da - sofern k=3 beibehalten werden soll - drei Teilmengen aus der Kombination resultieren müssen. Für die Kombination $B + A$ verbinden sich die Centroide C_{A_1} mit C_{B_1} sowie C_{B_2} mit C_{A_2} und C_{B_3}. Für Centroid C_{B_3} findet keine Verbindung statt. Die niedrigste Varianzsumme wird die auserwählte Kombination. Dieses Verfahren hat ferner den Vorteil, dass die bessere Kombination gewählt wird, was sich anschließend in weniger Iterationen des Algorithmus widerspiegelt. Somit amortisiert sich der Rechenaufwand, der durch die Berechnung der Varianzen nötig wurde.

Nachdem das Problem der Kommutativität durch ein zusätzliches Entscheidungsmerkmal gelöst wurde, steht ein weiteres noch bevor. Der Kombination mangelt es zudem an Assoziativität. Es wurde bereits erwähnt, dass bei der lokalen Gruppenbildung das Endergebnis von der Reihenfolge der Gruppierungsobjekte abhängt, doch auch die globale Gruppenbildung ist von diesem Phänomen betroffen. Dass die Reihenfolge der Verschmelzungen nicht willkürlich stattfinden darf, zeigt das Beispiel in Abbildung 4.38. In der Abbildung sind unterschiedliche Ergebnisse für die Verschmelzung von lokalen Gruppen zu sehen. Zurückzuführen ist dieses Resultat auf die Tatsache, dass die Suche des nächstgelegenen Centroid nicht assoziativ ist.

Dieses Problem kann nicht wie im Falle der Kommutativität gelöst werden, indem sämtliche Fälle untersucht werden und anschließend eine Möglichkeit gewählt wird. Bei der Kommutativität mussten nur zwei Fälle betrachtet werden, wogegen es bei der Assoziativität wesentlich mehr Fälle[17] sind und eine Überprüfung der einzelnen deshalb nicht durchführbar ist.

Die Kombination mittels des k-Means Algorithmus muss aus diesem Grund stets nach einer definierten Reihenfolge ablaufen. Als Vergleichsobjekte dienen die Identifikatoren, die in Abschnitt 4.1.2 eingeführt wurden. Ein globales Gruppieren kann somit für manche Knoten erst beginnen, wenn sämtliche lokalen Gruppen vorliegen, um die Reihenfolge einhalten zu können. Dies trifft nicht auf Knoten zu, die die ersten zu verknüpfenden Gruppen bereits erhalten haben. Die Identifikatoren sind durch die Segmentierung und Topologieerzeugung im Segment bekannt.

Die Kombination kann erst nach dem Austausch sämtlicher lokaler Gruppen erfolgen, d. h. um ein endgültiges Ergebnis zu bekommen, muss auf die letzte eintreffende lokale Gruppe gewartet werden.

Nachdem die Kombination für den k-Means Algorithmus ausführlich erläutert wurde,

[17] Für n Kombinationen gibt es insgesamt n! Möglichkeiten diese zu kombinieren. Durch die vorliegende Kommutativität wird diese zwar halbiert, was aber an der Komplexität nichts ändert.

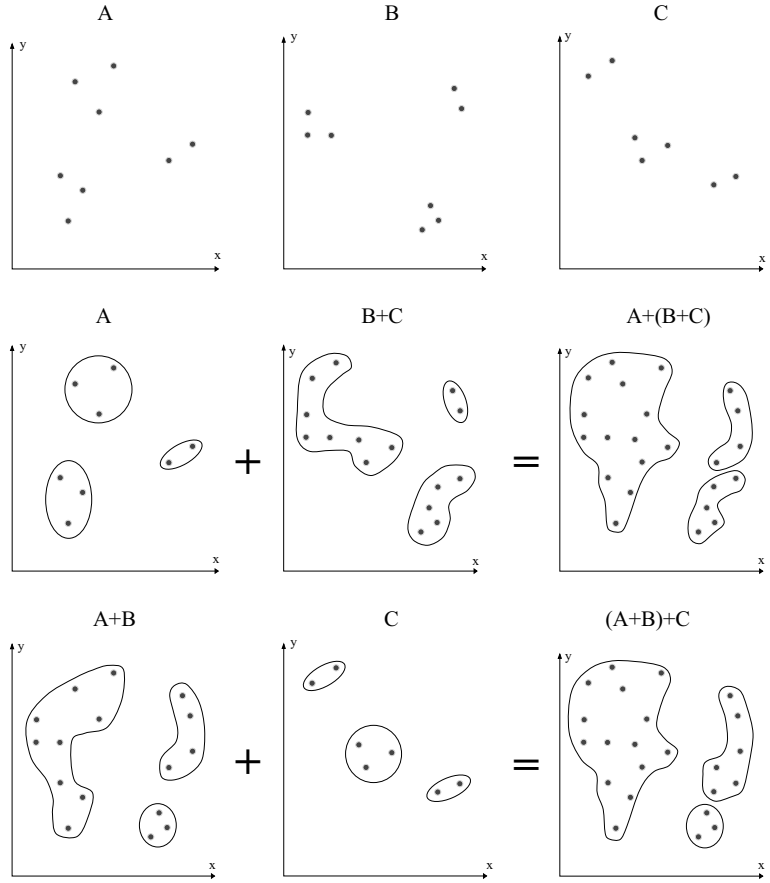


Abbildung 4.38: Die willkürliche Verschmelzung ist noch assoziativ.

kann die Beschreibung für den PAM-Algorithmus kürzer ausfallen, da einzelne Vorgehensweisen ähnlich sind.

Die Kombination zweier lokaler Gruppen geschieht wie beim k-Means Algorithmus, dargestellt in Abbildung 4.36, nur dass nicht die Centroide als Abstandsgrundlage dienen, sondern die Medoide. Obwohl bei PAM die Reihenfolge der Knoten für das erhaltene Endergebnis nicht relevant ist, wird trotzdem der Varianzvergleich gemäß Gleichung 4.35 durchgeführt, da, sofern durch diese Berechnung eine Iteration des PAM Algorithmus eingespart werden kann, sich dies in diesem Fall mehr als amortisiert und die Kosten im anderen Fall nur zwei schnell durchzuführende Berechnungen darstellen.

Im Unterschied zum k-Means muss beim PAM auch nicht auf die Reihenfolge geachtet werden, so dass die Berechnung starten kann, sobald neue Profilinformation eintrifft. Dies ist auch notwendig, da die Laufzeit der PAM quadratisch in n ist, wogegen sie beim iterativen k-Means Algorithmus linear ist. Die Zeit, die mit Warten auf neue Informationen verbracht wird, kann so sinnvoll für die Berechnung genutzt werden. Prinzipiell muss PAM mit sämtlichen Profilinformationen so oft iteriert werden, bis keine Vertauschung mehr durchgeführt wird. Erst damit liegt eine global gültige Gruppe vor. Was dies für die

Gruppengröße und den zeitlichen Aufwand bedeutet, zeigt Kapitel 6.

Um noch zusätzlich Rechenaufwand einzusparen, wird beim PAM Algorithmus zu den Gruppierungsdaten die Distanzmatrix übertragen, damit die Nachbarknoten diese nicht nochmals berechnen müssen.

Unterstützung der Mobilität: Nachdem der verteilte Gruppierungsvorgang beschrieben ist, muss noch geklärt werden, wie und unter welchen Umständen es möglich ist, nachdem bereits eine Globale Gruppe erzeugt wurde, noch zusätzlich Teilnehmer in die Gruppe aufzunehmen.

Solange der Gruppierungsprozess noch nicht abgeschlossen ist, können sich stets neue Teilnehmer anschließen. Hierfür gelten sowohl Gleichung 4.9 als auch Gleichung 4.10.

Ist jedoch schon eine globale Gruppe vorhanden und es wurden sämtliche Teilnehmer hiervon informiert, ist bei einer Hinzunahme eines neuen Teilnehmers nur Gleichung 4.9 zu berücksichtigen. Würde zusätzlich noch Gleichung 4.10 gefordert, so könnte die bereits vorhandene Gruppe wieder aufgeteilt werden müssen. Hierüber müssten wiederum sämtliche Teilnehmer informiert werden. Dies ist jedoch in den meisten Fällen nicht mehr möglich, weil sich diese Teilnehmer bereits außerhalb des Segments befinden. Aus diesem Grund ist es ausreichend, wenn sich nur der Nutzen durch neue Teilnehmer erhöht.

4.3.5 Gruppierungsablauf

Am Ende dieses Abschnittes soll der Gruppierungsablauf noch einmal zusammenfassend verdeutlicht werden. Das Ablaufdiagramm in Abbildung 4.39 informiert über die einzelnen

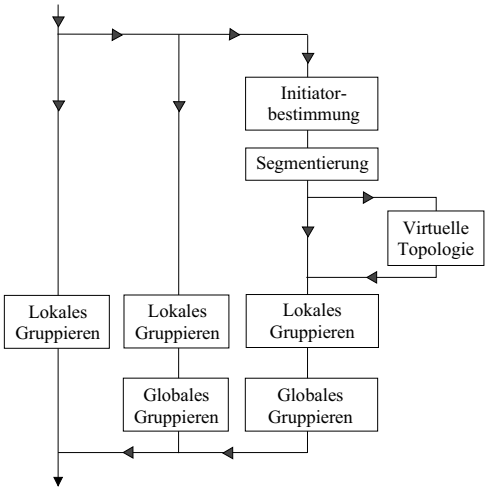


Abbildung 4.39: Möglichkeiten des Gruppierungsablaufes.

Möglichkeiten.

Die erste Sequenz stellt das Minimum an Aktionen dar, die nötig sind, um eine Gruppe zu erstellen. Hierzu reicht das Erstellen einer lokalen Gruppe aus, wenn ein quasi zentraler Koordinator über die Hopdistanz 1 eine Gruppe erzeugen will. Die erhaltene lokale Gruppe

wird zur globalen Gruppe, nachdem jeder in der lokalen Gruppe vorhandene Teilnehmer seine Teilnahme bestätigt hat.

Das mittlere Verfahren kommt zur Anwendung, wenn das ad hoc Netzwerk nicht zu groß ist, d. h. der Gesamtdurchmesser kleiner als der spezifizierte Segmentdurchmesser ist. In diesem Fall ist eine Segmentierung nicht notwendig und der Gruppierungsvorgang besteht nur aus lokalem und globalem Gruppieren.

In der rechten Sequenz in Abbildung 4.39 ist der vollständige Gruppierungsablauf aufgezeigt. Nach der Initiatorbestimmung wird mit Hilfe der Segmentierung das ad hoc Netzwerk in kleinere Teile aufgeteilt. Die Überlagerung mit einer virtuellen Topologie ist optional. Danach wird zuerst eine lokale Gruppierung durchgeführt, auf die eine globale Gruppenbildung folgt.

Abschließend sei gesagt, dass die einzelnen Teile im Ablauf gemäß Abbildung 4.39 nicht notwendigerweise stets nacheinander ausgeführt werden. Um Nachrichten einzusparen und außerdem zeitlich effizient zu sein, bietet es sich an, Aktionen parallel auszuführen. So kann beispielsweise die Segmentierung mit der Erstellung der virtuellen Topologie kombiniert werden.

4.4 Domänenunabhängigkeit von MoPiDiG

Die MoPiDiG Architektur, siehe Abbildung 4.1, besteht neben einer Algorithmuseinheit noch aus einem Meta-Beschreibungsmodul, welches die Domänenunabhängigkeit sicherstellt. Hier sind Meta-Beschreibungen enthalten, die Vorgaben enthalten, wie eine Profildefinition oder eine Gruppe aufgebaut sein muss, damit eine Gruppenbildung mit MoPiDiG durchführbar ist. Um zu überprüfen, ob eine Beschreibung den Anforderungen genügt, ist eine Konsistenzüberprüfung enthalten. Damit die Gruppenbildungsalgorithmen auf die Profileinträge zugreifen können, existiert ein Anfrage-Modul.

4.4.1 Meta-Beschreibungen

Die Gruppenbildungs-Algorithmen, vorgestellt in Abschnitt 4.3.4, müssen auf die Profildaten zugreifen können. Aus diesem Grund muss hierzu eine passende Schnittstelle existieren, die mit den Meta-Beschreibungen geschaffen wird.

Meta-Beschreibungen sind Schablonen für Benutzerprofile und Gruppendefinition, die in XML Schema [Fal01; TBMM01; BM01] definiert werden können. Somit kann ein Benutzerprofil bzw. eine Gruppendefinition in MoPiDiG nur verwendet werden, wenn beide den Meta-Beschreibungen entsprechen.

Nachfolgend werden die Meta-Profilbeschreibung sowie die Meta-Gruppenbeschreibung angegeben und deren Inhalt erläutert.

4.4.1.1 Meta-Profilbeschreibung

Die Meta-Profilbeschreibung beschreibt, wie die Struktur eines Benutzerprofils für MoPiDiG definiert ist. Nur wenn diese Struktur erfüllt ist, können die Gruppierungsalgorithmen die notwendigen Daten aus dem Profil extrahieren.

Der Aufbau eines Benutzerprofils ist in Abbildung 4.40 stark vereinfacht wiedergege-

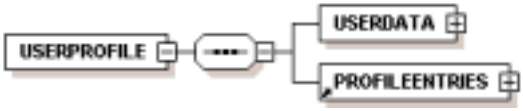


Abbildung 4.40: Grober Aufbau der Profilbeschreibung

ben. Wie hieraus zu ersehen ist, besteht ein `USERPROFILE` aus zwei Teilen. Zum einen ist hier der `USERDATA` Bereich zu nennen, der Daten über den Profilbesitzer enthält und zum anderen existiert ein `PROFILEENTRIES` Teil, das eine Aufzählung der Profileinträge darstellt.

Abbildung 4.41 zeigt die puristische Definition des `USERDATA` Bereichs. Wie zu sehen

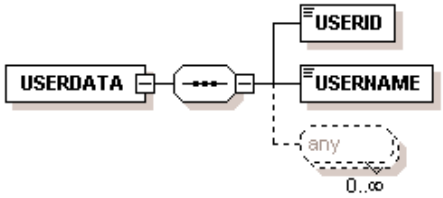


Abbildung 4.41: Benutzerdaten in der Profilbeschreibung

ist, besteht er nur aus zwei obligatorischen Elementen, einer `USERID` und einem `USERNAME`. Die `USERID` entspricht dem Identifikator, der für MoPiDiG benötigt wird, während der `USERNAME` eine Bezeichnung für den Benutzer darstellt.

Mit diesen beiden Einträgen ist jedoch i. A. die Beschreibung eines Benutzers noch nicht abgeschlossen. Es könnten noch Adressdaten, Telefonnummern etc. definiert werden. Diese Einträge werden jedoch nicht gefordert und können optional erfolgen. Dies wird in Abbildung 4.41 mit dem Feld `any` angedeutet, welches in beliebiger Anzahl auftreten darf. Zu beachten ist jedoch, dass, wenn zusätzlich Daten definiert werden, auch ein weiteres XML-Schema für die Zusatzdaten vorhanden sein muss, damit auch das Gesamtprofil gültig ist.

Der in Abbildung 4.40 zu sehende `PROFILEENTRIES` Eintrag besteht aus einer beliebigen Anzahl von `PROFILEENTRY` Einträgen, deren Aufbau in Abbildung 4.42 zu sehen ist. Ein `PROFILEENTRY` besteht aus einem Namen (`ENTRYNAME`) und einem Wert. Für diesen Wert kann zwischen einer Zeichenkette und einem Zahlenwert gewählt werden. Komplexe Datentypen werden momentan nicht unterstützt, da dies hohe Anforderungen an die Meta-Beschreibung stellen würde und für die Beispielszenarien in Kapitel 5 nicht benötigt werden.

Auf diese Weise wird ein Key-Value-Pair erhalten. Damit ferner noch die Möglichkeit besteht eine Hierarchie wie in Abbildung 3.6 (siehe Abschnitt 3.2.2) generieren zu können, ist noch ein optionaler `SUBENTRIES` Eintrag erlaubt.

Wie Abbildung 4.43 zeigt, bestehen die `SUBENTRIES` wieder aus einer Menge an `PRO-`

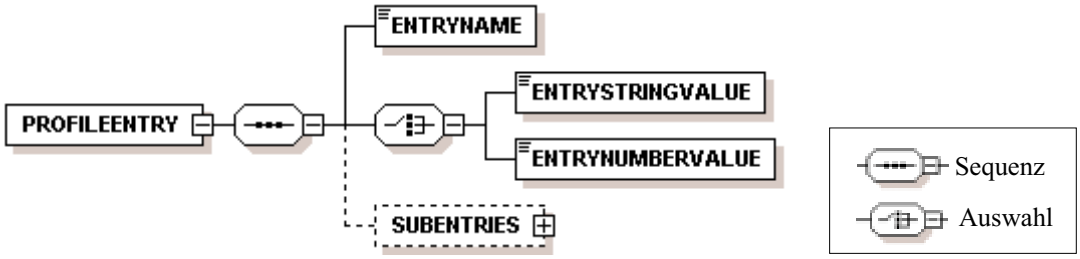


Abbildung 4.42: Diagram der `PROFILEENTRY` Profilbeschreibung.

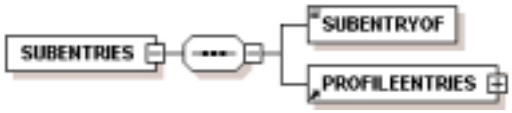


Abbildung 4.43: `SUBENTRIES` in der Profilbeschreibung

`FILEENTRIES`, womit eine Hierarchie mit einer beliebigen Anzahl an Stufen erreicht werden kann. Der zusätzliche Eintrag `SUBENTRYOF` spezifiziert den Vaterknoten. Liegt stets das komplette Benutzerprofil vor, könnte auf diesen Eintrag verzichtet werden, da jedoch auch Teile des Profils zur Gruppierung herangezogen werden können, erspart diese Angabe das sonst notwendige Wechseln in eine niedrigere Hierarchiestufe.

In Appendix A.1 ist das vollständige XML Schema für das Benutzerprofil angegeben.

4.4.1.2 Meta-Gruppenbeschreibung

Die Meta-Gruppenbeschreibung legt fest, wie eine Gruppendefinition in MoPiDiG auszusehen hat.

In MoPiDiG muss eine Gruppenbeschreibung Informationen über die zu gruppierenden Objekte sowie Angaben über die Gruppengröße und zeitliche Gültigkeit der Gruppen enthalten. Abbildung 4.44 zeigt den Aufbau der Meta-Gruppenbeschreibung. Eine `GROUPDESCRIPTION` hat für jede Gruppierungsart (`GROUPTYPE`) eine andere Struktur, da sich jeweils die zu gruppierenden Objekte (`ENTRIES`) unterscheiden.

Beim nutzenbasierten Gruppieren besteht der `ENTRIES` Eintrag nur aus einer Liste von `ENTRYNAME` Einträgen. Diese müssen definiert werden, weil sie für die Errechnung des Nutzens (Funktion π) von Bedeutung sind.

Beim explizit ähnlichkeitsbasierten Gruppieren werden Personen gesucht, die gleiche Profileinträge vorweisen können. Deshalb wird hier neben dem `ENTRYNAME` auch noch der Wert (`ENTRYSTRINGVALUE` bzw. `ENTRYNUMBERVALUE`) angegeben. Zusammengefasst werden diese in einem `ENTRY` Tag.

Das implizit ähnlichkeitsbasierte Gruppieren sucht Gruppen, die ähnliche Profileinträge haben, weshalb die Spezifikation von Profileinträgen nicht zwingend erforderlich ist.

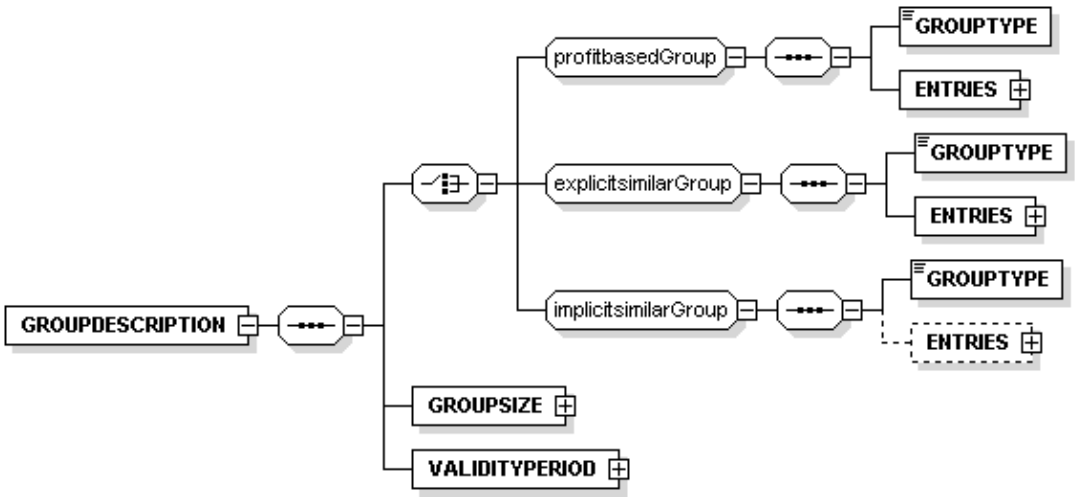


Abbildung 4.44: Struktur der Gruppenbeschreibung

Die Tatsache, dass jedoch ein `ENTRYNAME` angegeben werden kann, hat den Zweck, dass nicht das komplette Profil betrachtet werden muss, sondern hier nur ein Profilteil angegeben wird.

Abbildung 4.45 informiert über den Aufbau des `GROUPSIZE` Eintrags. Dieser besteht aus

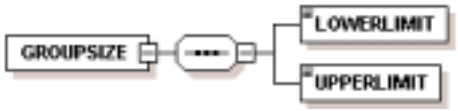


Abbildung 4.45: Definition der Gruppengröße in der Gruppenbeschreibung

der Angabe einer Ober- und Untergrenze für die Gruppengröße. Die zeitliche Gültigkeit besteht nur in der Spezifikation eines Start- und eines Zielzeitpunktes, wie in Abbildung 4.46 zu sehen ist, um festzulegen, wann eine Gruppenbildung durchgeführt werden soll.

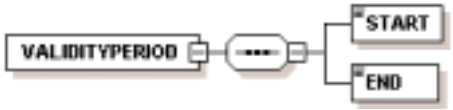


Abbildung 4.46: Modellierung der Gültigkeitsdauer der Gruppenbeschreibung

Im Appendix A.2 ist das komplette XML Schema für die Gruppendefinition angegeben.

4.4.2 Zugriff auf das Profil sowie Profilüberprüfung

Nachdem die Schablonen für das Benutzerprofil und der Gruppenbeschreibung vorgestellt wurden, werden in diesem Abschnitt die Methoden präsentiert, die den Zugriff auf das Benutzerprofil erlauben und ein Benutzerprofil auf Gültigkeit überprüfen.

4.4.2.1 Profilzugriff

Oft ist es notwendig zu überprüfen, ob gewisse Einträge in einem Benutzerprofil vorhanden sind. Dies ist beispielsweise für das explizite ähnlichkeitsbasierte Gruppieren der Fall.

Um im Benutzerprofil nach einem oder mehreren Einträgen zu suchen wird XPath verwendet. XPath [CD99] ist eine Abfrage-Sprache, um Teile eines XML-Dokumentes zu adressieren. XPath stellt Methoden zur Verfügung, mit denen es auf einfache Weise möglich ist, beliebige Knoten aus dem XML Dokument zu extrahieren. Mit XPath ist es auch möglich mehrere Anfragen mit einer Anweisung zu erstellen.

4.4.2.2 Profilüberprüfung

Dieser Abschnitt beschreibt das existierende Verfahren, welches feststellt, ob eine Profildefinition der Meta-Profilbeschreibung und eine Gruppendefinition der Meta-Gruppenbeschreibung entspricht. Ferner dürfen in Gruppendefinitionen nur Attributnamen von Profileinträgen existieren, die auch in der Profildefinition vorhanden sind.

Um zu überprüfen, ob sowohl die Profildefinition als auch die Gruppendefinition den entsprechenden Meta-Beschreibungen genügen, wird ein XML Schema Validator verwendet, z. B. Multi-Schema-Validator [Mic03b] oder XML Schema Validator [W3C03]. Dieser kontrolliert in einem ersten Schritt, ob das XML Dokument wohlgeformt ist, d. h. ob die Syntax korrekt ist. Danach wird überprüft, ob sämtliche gemäß dem XML Schema geforderte Elemente vorhanden sind, die Anordnung korrekt ist und auch keine zusätzlichen Einträge existieren.

Mit den XML Schema Validator ist es jedoch nicht möglich, die inhaltliche Korrektheit festzustellen, so dass diese noch zusätzlich durchzuführen ist. So muss in der Gruppendefinition überprüft werden, ob die Einträge für die Gruppengröße korrekt sind. Der Eintrag für `UPPERLIMIT` darf den Eintrag in `LOWERLIMIT` nicht übersteigen um verwendet werden zu können. Ähnlich ist es mit den Einträgen für die zeitliche Gültigkeit. Hier muss kontrolliert werden, ob der `START` tatsächlich vor dem `END` Eintrag liegt. Ansonsten würde die Gruppenbeschreibung zu keiner Zeit Gültigkeit aufweisen.

Schließlich muss noch kontrolliert werden, ob die Gruppendefinition mit dem Benutzerprofil harmoniert. Die in der Gruppendefinition angegebenen Einträge, die für die Gruppierung notwendig sind, müssen im Benutzerprofil vorhanden sein. Ansonsten kann keine Gruppierung stattfinden. Die Überprüfung kann zurückgeführt werden auf eine Profilanfrage mit XPath [CD99], wie im vorherigen Abschnitt geschildert. Sämtliche Elemente, nach denen gruppiert werden soll, werden zuerst mit XPath aus der Gruppendefinition extrahiert. Danach wird mit jedem dieser Einträge eine Anfrage an das Benutzerprofil gestellt. Nur wenn sämtliche Einträge vorhanden sind, ist die Verifikation abgeschlossen.

4.4.3 Domänenbeschreibung

In der Domänenbeschreibung sind Instanzen der Meta-Profilbeschreibung und Meta-Gruppenbeschreibung enthalten. Ferner existiert noch eine Domänendefinition, in der die Konfiguration der Anwendung stattfindet.

4.4.3.1 Profildefinition

Eine Profildefinition erfolgt in XML (eXtensible Markup Language) [BPMY04; BPSM+04] und muss dem in Abschnitt 4.4.1.1 definierten Schema entsprechen. Die Überprüfung übernimmt gemäß Abschnitt 4.4.2.2 der Validator.

Neben gewöhnlichem XML hätte sich zur Profilbeschreibung auch OWL (Web Ontology Language) [MvH04] angeboten. Mit OWL ist es möglich semantische Vokabulare (Ontologien) zu definieren. OWL existiert in drei Ausführungen; OWL Light, OWL DL und OWL Full, wobei nur OWL Light und DL entscheidbar sind und damit von einer Reasoning Engine verarbeitet werden können. Für den Fall von Profildefinitionen (Taxonomien) würde OWL Light ausreichen und es wäre möglich aus vorhandenem Wissen neues zu generieren. Die Validierung von OWL Dokumenten ist wesentlich komplexer als die von Standard XML Dokumenten. Auch die Angabe eines Schemas erweist sich als ausgesprochen schwierig.

4.4.3.2 Gruppendefinition

Die Gruppendefinition ist ebenfalls in XML angegeben und muss der Meta-Gruppenbeschreibung (siehe Abschnitt 4.4.1.2) genügen, d. h. die Gruppendefinition ist eine Instanz der Meta-Gruppenbeschreibung.

In der Gruppendefinition sind Profileinträge vorhanden, auf deren Basis die Gruppenbildung durchgeführt wird. Diese Profileinträge müssen Teil des Benutzerprofils sein. Untersucht wird dies von der Konsistenzüberprüfungskomponente.

4.4.3.3 Domänendefinition

In der Domänendefinition erfolgt die Konfiguration der eigentlichen Anwendung. Die Konfiguration erfolgt mit einem XML-Dokument und mehreren Java Klassen und Interfaces, die implementiert werden können bzw. müssen.

In dem XML-Dokument der Domänendefinition kann bestimmt werden, ob eine und wenn ja welche virtuelle Topologie zu verwenden ist. Außerdem kann der Parameter k zur Segmentgrößenbestimmung hier angegeben werden. Hinsichtlich der Initiatorbestimmung kann zwischen dem zufallsbasierten und identifikatorbasierten Ansatz gewählt werden. Ferner wurden in MoPiDiG bereits sämtliche in diesem Kapitel erwähnten Distanzfunktionen integriert, deren Anwendung in der Konfiguration angegeben werden kann.

Soll jedoch eine Distanzfunktion verwendet werden, die noch nicht definiert ist, muss diese in einer neuen Java Klasse zur Verfügung gestellt werden. Auch die Payoff-Funktion π für das nutzenbasierte Gruppieren muss auf diese Weise angegeben werden.

4.5 Zusammenfassung

In diesem Kapitel steht die Architektur des MoPiDiG Frameworks im Mittelpunkt. Zuerst wird die Architektur vorgestellt.

Das MoPiDiG Framwork ist modular aufgebaut und besteht aus drei Ebenen. Grundlage ist eine Middleware, die Kommunikationsmöglichkeiten schafft. Eine weitere MoPiDiG Ebene beinhaltet die Algorithmen und Meta-Beschreibungen um die Domänenunabhängigkeit zu erreichen. Die oberste Ebene dient dazu, das MoPiDiG Framework an eine spezielle Anwendung anzupassen.

Bezüglich der Algorithmen reicht es nicht aus, sich nur auf die Gruppierung zu konzentrieren. Da ad hoc Netzwerke aus einer großen Anzahl an Teilnehmern bestehen können, muss eine Methode vorhanden sein, die ein großes Netzwerk in mehrere kleine aufteilt. Damit diese Segmentierung nicht von allen Teilnehmern gleichzeitig vollzogen wird, ist eine Initiatorbestimmung notwendig, die die initiierenden Teilnehmer festlegt. Um die Anzahl an Nachrichten zu reduzieren, kann, sofern es das Geschwindigkeitsprofil erlaubt, eine virtuelle Topologie kreiert werden. Als Topologien werden eine Baumstruktur und die Ringtopologie diskutiert.

In MoPiDiG sind zwei Gruppierungsverfahren vorhanden. Im nutzenbasierten Gruppierungsverfahren ist jeder Teilnehmer mit einer Nutzenfunktion ausgestattet. Es schließen sich nur Objekte zu Gruppen zusammen, wenn sich dadurch bei jedem Teilnehmer die Nutzenfunktion erhöht. Beim ähnlichkeitsbasierten Gruppierungsverfahren werden aus der Menge der möglichen Gruppierungsteilnehmer, die, gemäß einer definierten Distanzfunktion ähnlichsten Objekte, zu einer Gruppe vereinigt.

Die Gruppierung wird in zwei Schritten vollzogen. Im ersten Schritt wird eine lokale Gruppe erzeugt, die aus einer Teilmenge der direkten Nachbarn besteht. Diese lokalen Gruppen werden im zweiten Schritt ausgetauscht, bis schließlich eine globale Gruppe vorhanden ist.

Nachdem somit der allgemeine MoPiDiG Aufbau bekannt ist, wird im nächsten Kapitel die Implementierung eines Beispielszenarios erläutert.

Kapitel 5

Anwendung des MoPiDiG Frameworks

In diesem Kapitel wird eine konkrete Anwendung des MoPiDiG Frameworks näher beschrieben. Als Beispiel wird ein Taxi-Sharing-Szenario vorgestellt, welches im ersten Abschnitt erklärt wird. Danach erfolgt eine Beschreibung existierender Middleware, die prinzipiell für ad hoc Netzwerke geeignet ist.

Im nächsten Abschnitt werden die formalen Grundlagen für das Taxi-Sharing Szenario geschaffen. Neben notwendigen Definitionen werden hier auch Varianten präsentiert, wie im Taxi-Sharing-Szenario die Fahrpreisbildung von statten geht.

Nachdem das Taxi-Sharing-Szenario formal definiert wurde, wird danach der Gruppierungsprozess erläutert. Hier wird sowohl das lokale Gruppieren, als auch die globale Gruppenbildung im Detail beschreiben. Außerdem werden mehrere Heuristiken vorgestellt, die den Gruppierungsprozess noch effizienter gestalten.

Im nächsten Abschnitt wird die domänenunabhängige Schicht erklärt. Hier sind vor allem die Gruppen- und Profildefinitionen für die konkrete Anwendung angegeben.

Danach werden die Simulationsumgebung und das zu Grunde liegende Bewegungsmodell erläutert.

Im letzten Abschnitt werden anhand eines weiteren Beispiels die Gruppierungsarten vorgestellt, die für das Taxi-Sharing-Szenario nicht von Belang sind.

Am Ende des Kapitels ist eine kurze Zusammenfassung zu finden.

5.1 Taxi-Sharing-Szenario

Das Taxi-Sharing Szenario wurde bereits in Abschnitt 1.1.2.1 kurz erklärt. Ausgangspunkt hierfür ist ein stark frequentierter Ort, wie z. B. der in Abbildung 5.1 dargestellte Bahnhof. Von jedem Teilnehmer wird gefordert, dass er ein mobiles Endgerät[1] besitzt, auf welchem sein Benutzerprofil gespeichert ist. Als Profilinformation ist der Zielort der Person gegeben. In der vorliegenden Implementierung entspricht dies einer x-y-Koordinate. Symbolische Namen wie Straßennamen können nicht verwendet werden, da gemäß den Annahmen keine Ortsinformation vorhanden ist. Zusätzliche Daten wie zeitliche Einschränkungen

[1] In der Implementierung werden als Endgeräte PDAs verwendet. Mobiltelefone sind in ihrer derzeitigen Ausstattung noch nicht für diesen Einsatz geeignet.

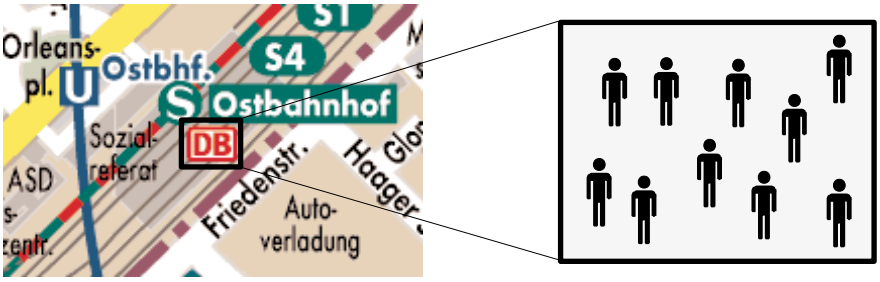


Abbildung 5.1: Taxi-Sharing-Szenario

sind fakultativ. Die Gruppengröße beläuft sich auf zwei bis vier Teilnehmer, weil dies der räumlichen Kapazität eines Taxis entspricht.

Während die Personen sich im Zug oder aber auch im Bahnhofsgebäude befinden, wird die Profilinformation ausgetauscht und auf diese Weise versucht eine oder mehrere Gruppen zu erzeugen. Zu einer Gruppe schließen sich Teilnehmer zusammen, wenn sie entweder das gleiche Ziel haben oder das Ziel ohne Umwege zu erreichen ist. Grundvoraussetzung für die Gruppenbildung ist, dass sich der Fahrpreis für jede teilnehmende Person im Vergleich zur Einzelfahrt reduziert.

Abbildung 5.2 zeigt das Ergebnis einer Gruppierung. Die Punkte verdeutlichen die

Abbildung 5.2: Routenentwicklung beim Taxi-Sharing. Die Punkte symbolisieren die Ziele der Gruppierungsteilnehmer. Die zwei Linien stellen zwei Taxi-Routen dar. Wie zu sehen ist existieren zudem Punkte die nicht Teil einer Route sind.

Zielorte der Teilnehmer. Teilnehmer, die zu einer Gruppe gehören, sind durch eine entsprechende Linie verbunden. Es müssen nicht alle Teilnehmer einer Gruppe zugeordnet werden. Zum einem kann der Zielort zu weit von einer Route entfernt sein und zum anderen kann die Gruppengröße von vier Personen bereits überschritten sein.

Nachdem eine Gruppe gebildet wurde, müssen sich die Teilnehmer noch zusammenfinden, d. h. ein Treffpunkt ist notwendig. Durch die Übermittlung von Mobilfunknummern kann dieser telefonisch vereinbart werden.

5.2 Middleware für Mobile ad hoc Netze

Nachdem im letzten Abschnitt die Taxi-Sharing Applikation beschrieben wurde, wird nachfolgend die Architektur vorgestellt, wie sie in Abschnitt 4.1 präsentiert wurde. Die Middleware stellt den Anfang der Beschreibung dar.

Die Middleware ist die unterste Ebene (siehe Abbildung 4.1 in Abschnitt 4.1) einer MoPiDiG Anwendung, wobei die Existenz passender Middleware als gegeben vorausgesetzt wird. Dies bedeutet jedoch, dass existierende Middleware auf deren Anwendbarkeit in mobilen ad hoc Netzen zu untersuchen ist.

In den letzten Jahren wurden mehrere Frameworks entwickelt, die prinzipiell als Middleware für ad hoc Netzwerke eingesetzt werden können. Eine optimale Lösung, die gänzlich von der ad hoc Vernetzung der Geräte abstrahiert und ferner für alle Arten von Geräten - von Laptop bis Smartphone - verfügbar ist, existiert derzeit noch nicht.

Nachfolgend werden die Frameworks beschrieben, die primär Verwendung finden. Abbildung 5.3 zeigt vorab einen Überblick bezüglich Umfang und Möglichkeiten der Frameworks.

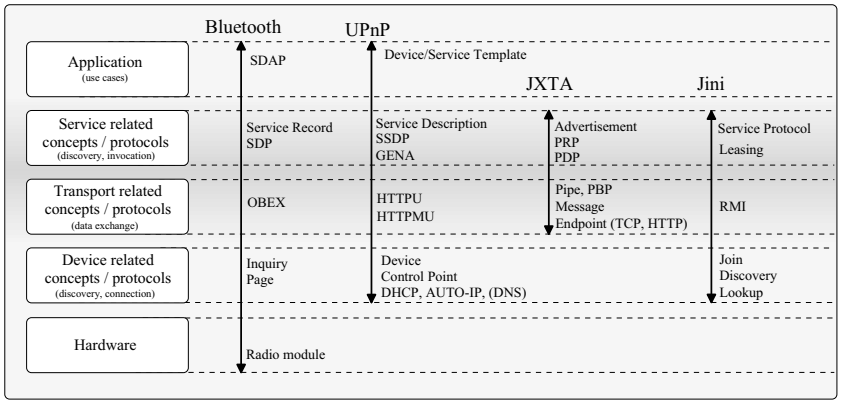


Abbildung 5.3: Middleware in Anlehnung an Pirker [Pir03]

5.2.1 Jini

Jini [Sun03a] war der erste Standard (ein Satz von APIs) für die spontane Kommunikation von Geräten, weswegen er auch hier kurz Erwähnung finden soll. Jini-Netze kommen vollständig ohne zentrale Steuerung aus, sind unabhängig von Betriebssystemen und Netzwerkstrukturen und sollen eine einfache und direkte Nutzung aller angeschlossenen Geräte ermöglichen. In einem Jini-Netzwerk können neue Teilnehmer hinzukommen oder sich ent-

fernen ohne den zusätzlichen Eingriff von außen. Die Kommunikation in Jini erfolgt über RMI[2].

Die Geräte, Daten und Services[3] innerhalb des Netzwerkes werden über einen „*Lookup Service*“ bereitgehalten. Zu ihm sucht jedes neu angeschlossene Gerät automatisch Kontakt. Ein *Leasing*-Mechanismus sorgt dafür, dass das Jini-Netzwerk stets aktuell ist. Jeder Client muss permanent sein Interesse an der weiteren Nutzung eines Services bekunden, ansonsten wird er als nicht mehr vorhanden angenommen.

Da der *Lookup Service* jedoch als zentrales Element realisiert ist, würde die Anwendung von Jini als Middleware für MoPiDiG stark einschränken.

5.2.2 UPnP

Wie bei Jini sollen auch bei Universal Plug and Play (kurz UPnP) [Mic00a] verschiedenartige Geräte miteinander kommunizieren können. UPnP stellt Standards von Microsoft für Netzwerkumgebungen dar, die es ermöglichen, dass sich Geräte universell vernetzen sowie untereinander Dienstleistungen austauschen - ohne zentralen Server. UPnP ist von keinem bestimmten Betriebssystem, keiner Programmiersprache, keinem physikalischen Medium abhängig und es können alle physikalischen Medien, die IP-Kommunikation unterstützen, verwendet werden, z. B. Ethernet, Bluetooth oder Wireless LAN. Die Technik basiert im Wesentlichen auf offenen Standards und standardisierten Technologien, wie IP, UDP Multicast, TCP, HTTP, SOAL und XML. Hierdurch wird eine Kompatibilität zu jedem Netzwerk erreicht.

Bei UPnP heißt ein Knoten in der Rolle des Dienstnutzers *Control Point* und ein Dienstanbieter (*Controlled) Device*, wobei jeder Knoten auch beide Funktionen gleichzeitig in sich vereinen kann.

Da die Basis von UPnP ein IP-Netzwerk ist, muss ein Gerät oder *Control Point* zuerst über eine gültige IP-Adresse verfügen. Dies kann nach dem UPnP-Standard einerseits via DHCP[4] erfolgen oder via AUTO-IP [Che00]. Sobald ein UPnP-Gerät über eine IP-Adresse verfügt, muss es seine Existenz im Netzwerk an die *Control Points* melden. Das erfolgt via UDP[5] auf der Basis von SSDP[6]. Auf diese Weise wird auch die Suche nach UPnP-Geräten im Netzwerk von *Control Points* realisiert.

Nachdem ein *Control Point* ein Gerät gefunden hat, holt sich dieses per HTTP über TCP/IP die Beschreibung des Gerätes von der URL, welche ihm bei der Lokalisierung mitgeteilt wurde. Diese stellt das Gerät in Form eines XML-Dokumentes zur Verfügung. Die Beschreibung beinhaltet Informationen über den Hersteller, die Seriennummer, URLs für die Steuerung, Ereignisse und die Präsentation [Mic00b].

5.2.3 JXTA

JXTA[7] [Gon01] bildet eine grundlegende Netzwerk Infrastruktur, um einen einfachen, kleinen und flexiblen Mechanismus zu erhalten, der es erlaubt Peer-to-Peer Netze ad hoc

[2] Remote Method Invocation
[3] Ein Service ist der Anbieter einer speziellen Leistung im Netzwerk.
[4] Dynamic Host Configuration Protocol [Dro97].
[5] Multicast-Adresse 239.255.255.250:1900
[6] Simple Service Discovery Protocol
[7] Abkürzung für juxtapose (engl.) $\hat{=}$ nebeneinander stellen.

zur Verfügung zu stellen. Bei JXTA handelt es sich um eine Menge von Protokoll-Spezifikationen, wobei von vorhandenen Transportprotokollen wie TCP, UDP, HTTP etc. abstrahiert wird.

In JXTA sind sechs Protokolle [Sun03b] spezifiziert. Eine Interaktion zwischen Peers kann nur erfolgen, wenn alle Peers die gleichen Protokolle benutzen. Das *Peer Resolver Protocol* ist für das Auffinden von Peers zuständig. Über das *Peer Discovery Protocol* bieten die Peers ihre Dienste an und suchen nach den Diensten der anderen Peers. Statusinformationen über die Peers können mit dem *Peer Information Protocol* verbreitet werden. Das *Pipe Binding Protocol* stellt die Pipes für die Kommunikation zur Verfügung, dabei sorgt das Endpoint Routing Protocol dafür, dass zwischen den beiden beteiligten Peers eine Route auch mit Hilfe anderer Peers gefunden werden kann.

Damit die vorgenannten Protokolle ihre Informationen an alle beteiligten Peers verteilen können, müssen diese sich über einen Rendevous-Server bekannt machen. Dieser steuert die Informationen, die an alle Peers versendet werden sollen. Dazu nutzt er das *Rendezvous Protocol*.

Die Verbreitung der Information über die Existenz sämtlicher dieser in JXTA existierenden Entitäten wird durch so genannte Advertisements gehandhabt. Advertisements sind ressourcenbeschreibende Metadaten. Als Format für die Advertisements sowie für die Übertragung von Daten zwischen einzelnen Peers wird in JXTA der Standard XML verwendet.

Für JXTA existiert eine Referenzimplementierung für J2SE. Für J2ME[8] [Sun02] existiert eine JXTA-Implementierung, die unter dem Namen JXME [Mic03a] bekannt ist. Die Problematik ist jedoch, dass sich beide Varianten sehr unterscheiden. So ist in JXME ein spezieller *Proxy-Service* notwendig, der als Einstiegspunkt für JXME-Peers dient. Durch diese zusätzliche Infrastruktur ist JXME für ad hoc Netzwerke nur bedingt geeignet. Es sind jedoch Unternehmungen erkennbar, die diese Unterschiede ausräumen sollen.

5.2.4 Bluetooth

Auch Bluetooth bietet die Möglichkeit, dass Geräte untereinander ad hoc kommunizieren können. In Bluetooth sind außer der Funkschnittstelle ein kompletter Protokoll-Stack [Blu01a] und so genannte Profile [Blu01b] definiert. Durch die Profile ist es möglich, unterschiedliche Geräte ohne Konfiguration durch den Benutzer miteinander kommunizieren zu lassen. Wenn eine Bluetooth Verbindung aufgebaut wird, tauschen die Systeme ihre Profile aus und legen damit fest, welche Dienste sie für die jeweiligen anderen Partner zur Verfügung stellen können und welche Daten oder Befehle sie dazu benötigen.

Die einzelnen Profile sind nicht Fokus dieses Abschnitts, doch sollen zwei Profile nicht unerwähnt bleiben, weil mit diesen die ad hoc Funktionalität erreicht werden kann. Das *Generic Access Profile* (GAP) stellt die Basis aller Bluetooth-Profile dar und ist in jedem Bluetooth-Gerät enthalten. Über das GAP erkennen sich die Bluetooth-Geräte gegenseitig, fragen sich, ob sie miteinander reden dürfen und sichern dann die Übertragung gegenüber Dritten.

Die Geräteinformationen, Dienste und Leistungsmerkmale aller Stationen werden mit

[8] Java 2 Platform Micro-Edition. Umsetzung der Programmiersprache Java für stark ressourcenbeschränkte Endgeräte, wie etwa Mobiltelefone.

dem *Service Discovery Protocol* (SDP) abgefragt. Es ist ferner für die Diensterkennung der Gegenstelle notwendig. Sobald der Benutzer verfügbare Dienste in seiner Umgebung entdeckt hat, kann er eine Verbindung zwischen zwei oder mehreren Geräten aufbauen.

Für Bluetooth existiert eine J2ME-API, die im JSR 82[9] [Mot02] spezifiziert ist. In dieser API sind die Methoden zur Erkennung von Geräten und zum Verbindungsaufbau sowie wesentliche Profile enthalten.

Der Einsatz von Bluetooth ist jedoch für die Gruppierungsanwendung dahingehend eingeschränkt, da ein Piconetz aus maximal acht Geräten besteht (siehe hierzu auch Abschnitt 3.1.4.2) und somit wenige Teilnehmer zur Gruppenbildung zur Verfügung stehen. Die Zusammenführung mehrerer Piconetze zu einem Scatternet würde dieser Einschränkung entgegen wirken, doch sind Scatternetze derzeit noch aktiver Forschungsgegenstand.

5.2.5 Verwendete Middleware

Kernstück des verwendeten Systems bildet die Agentenplattform LEAP[10]. Da LEAP in seiner ursprünglichen Form keine ad hoc Funktionalitäten aufweist, wurde sie von Pirker [Pir03; PBW04] entsprechend erweitert. Realisiert wurde dies durch eine auf JXTA basierende ad hoc Komponente, die zur Geräteerkennung (device discovery) dient und diese Daten der Agentenplattform zur Verfügung stellt. JXTA wurde gewählt, da JXTA-Advertisements sich am besten zur Beschreibung in LEAP-Ressourcen eignen.

Für die Einbettung der ad hoc Komponente in eine Agentenplattform sprechen Interoperabilitätsgründe. Da die Agentenplattform LEAP FIPA[11] konform ist und die ad hoc Erweiterung ebenso diesem Standard Folge leistet, können sämtliche existierende FIPA-Plattformen als Middleware dienen und miteinander kommunizieren.

Prinzipiell kann mit wenigen Änderungen auch nur JXTA verwendet und auf die Agentenplattform verzichtet werden - zu Lasten der Interoperabilität. Obwohl Interoperabilität grundsätzlich erwünscht ist, sprechen Leistungsaspekte gegen den dualen Ansatz. Die Kombination LEAP-JXTA führt dazu, dass Nachrichten beide Frameworks durchlaufen und zudem noch in die jeweilig andere Datenrepräsentation konvertiert werden müssen.

5.3 Formalisierung des Taxi-Sharing Szenarios

Bevor der Gruppierungsvorgang in MoPiDiG erläutert wird, sind noch einige formale Angaben notwendig. Neben Definitionen werden ebenfalls verschiedene Varianten der Fahrpreisfestlegung besprochen.

5.3.1 Definitionen

Nachfolgend sollen die Definitionen aus Abschnitt 4.3.4.1 am Beispiel der Taxi-Sharing Applikation verdeutlicht werden.

[9] Java Service Request
[10] Lightweight Extensible Agent Platform
[11] Foundations for Intelligent Physical Agents: Internationales Konsortium aus industriellen und akademischen Organisationen mit dem Ziel einen einheitlichen Standard für Agentensysteme zu schaffen [FIP03].

Profilraum Π:

Der Profilraum $\Pi = \Pi_1 \times \Pi_2$ wird mit $\Pi_1 = \Pi_2 = [0..100]_{\mathbb{R}}$ definiert. Die Abstandsfunktion ρ wird gemäß des Euklidischen Abstandes definiert, d. h. der Abstand zweier Profile $P_i = (p_1^i; p_2^i)$ und $P_j = (p_1^j; p_2^j)$ beträgt

$$\rho(P_i, P_j) = \sqrt{(p_1^i - p_1^j)^2 + (p_2^i - p_2^j)^2}$$

Payoff Funktion π:

Beim Taxi-Sharing werden dann und nur dann Gruppen gebildet, wenn sich für die Gruppenmitglieder ein Nutzen abzeichnet. Eine Möglichkeit, den Nutzen festzustellen, ist die Betrachtung des Fahrpreises.

Entspricht P_S^i dem Preis, den der Teilnehmer i zahlen muss, wenn er in einer Einzelfahrt ans Ziel kommen muss und entspricht $P_{\mathcal{G}}^i$ den Kosten des Teilnehmers, wenn er Mitglied der Gruppe $\mathcal{G}$ ist, dann entspricht π:

$$\pi(i) = P_S^i - P_{\mathcal{G}}^i \tag{5.1}$$

$$\pi(\mathcal{G}) = \sum_i \pi(i) \tag{5.2}$$

Gleichung 5.1 entspricht einer Nutzenfunktion, da $\pi(i)$ ansteigt, wenn der Gruppenfahrpreis $P_{\mathcal{G}}^i$ für das einzelne Gruppenmitglied i geringer wird. Der Gesamtnutzen einer Gruppe ergibt sich gemäß Gleichung 5.2 als Summe aller Individualprofite $\pi(i)$.

Gruppierungsbedingung:

Generell wird eine Gruppe gebildet, wenn ein Nutzen feststellbar ist. Das bedeutet, dass

$$\pi(i) > 0 \tag{5.3}$$

gelten muss. Gleichung 5.3 kann auch durch den zu zahlenden Preis ausgedrückt werden.

Nachfolgend sei $P_{\mathcal{G}}$ der Preis, den die Gruppe $\mathcal{G}$ als Summe zahlen müsste. P_S^i entspricht wieder dem Preis, den der Teilnehmer i zahlen muss, wenn er in einer Einzelfahrt ans Ziel kommen muss. Aus Gleichung 5.3 wird somit

$$\forall i \in \mathcal{G} : P_S^i > P_{\mathcal{G}}^i \tag{5.4}$$

Gleichung 5.4 kann aufsummiert werden und da $P_{\mathcal{G}}$ der Summe aller Kosten $P_{\mathcal{G}}^i$ entspricht, folgt schließlich

$$\sum_i P_S^i > P_{\mathcal{G}}. \tag{5.5}$$

Gleichung 5.5 legt fest, dass der Gesamtfahrpreis $P_{\mathcal{G}}$ kleiner sein muss als die Summe aller Einzelfahrpreise $P_{\mathcal{G}}^i$. Während Gleichung 5.4 fordert, dass der Einzelpreis in der Gruppe gegenüber der Einzelfahrt abnehmen muss.

Somit sind zwei Bedingungen bekannt, die erfüllt sein müssen, damit im Taxi-Sharing-Szenario eine Gruppenbildung erfolgt.

5.3.2 Preisbildung und Gruppierungsergebnis

Die Tatsache, dass sich mehrere Personen ein Taxi teilen und dass sich hierdurch für jeden Mitfahrer der Fahrpreis im Gegensatz zur Einzelfahrt verringert, sagt nichts über die endgültige Preisverteilung der jeweiligen Gruppenmitglieder aus.

So wurde im vorherigen Abschnitt der Gruppenpreis $P_{\mathcal{G}}$ eingeführt. Es wurde jedoch nicht erwähnt, wie sich aus $P_{\mathcal{G}}$ die jeweiligen $P_{\mathcal{G}}^i$ für die Teilnehmer i ergeben.

Es gibt viele Möglichkeiten, wie der Gesamtfahrpreis am Ende unter den einzelnen Teilnehmern aufgeteilt wird. Die nachfolgenden drei Beispiele stellen nur eine Auswahl dar.

1. Gleiche Fahrtkosten: Der komplette Fahrpreis wird durch die Anzahl der Mitfahrer aufgeteilt, d. h. jeder zahlt den gleichen Betrag.
2. Anteilsmäßige Fahrtkosten: Gemeinsam gefahrene Strecken werden gleichmäßig unter den Gruppenteilnehmern aufgeteilt.
3. Gleiche prozentuale Ersparnis: Jeder Mitfahrer soll die gleiche prozentuale Ersparnis realisieren, d. h. wenn Mitfahrer A auf seiner Teilstrecke 20 % spart, so soll dies auch für Mitfahrer B gelten.

Wenn die Varianten hinsichtlich ihrer Gerechtigkeit bewertet werden, muss zuerst an der ersten Variante Kritik geübt werden, da bei dieser Methode jeder Fahrgast - unabhängig von der von ihm zurückgelegten Distanz - den gleichen Fahrpreis zahlen muss. Trotz dieser ungleichen Preisverteilung kann argumentiert werden, dass der Fahrpreis immer noch kleiner ist als für eine Einzelfahrtfahrt.

Bei Variante zwei und drei kann eher von einer gerechten Verteilung des Preises gesprochen werden. Bei Variante zwei wird nur für Strecken gezahlt, die auch zurückgelegt worden sind und ist deshalb aus der Sicht des einzelnen Fahrers am gerechtesten. Da bei Variante drei jeder den gleichen Nutzen hat, ist diese Methode aus Sicht der Gruppe am adäquatesten.

Die Gruppenbildung ist abhängig von der verwendeten Variante und bestimmt, wann jeweils eine Fahrgemeinschaft zu Stande kommt. Eine Gruppe wird nur gebildet, wenn der Einzelfahrpreis größer ist als der Fahrpreis, der sich durch die Gruppenbildung ergibt. Da für jede der Varianten der Gruppenpreis pro Teilnehmer auf eine andere Art berechnet wird, können sich andere Gruppen ergeben.

5.3.2.1 Variante 1: Gleiche Fahrtkosten

Bei dieser Variante wird der komplette Fahrpreis durch die Anzahl der Mitfahrer geteilt. Zwei Personen teilen sich ein Taxi, wobei eine Person den Ort A, die andere den Ort B als Ziel hat, wie Abbildung 5.4 schematisch verdeutlicht. Der Preis wird pro Kilometer berechnet und mit p_k bezeichnet[12].

Bilden beide keine Gruppe, so entspricht dies folgenden Preisen:

$$\text{Preis zu Ziel A: } P_A = a \cdot p_k \tag{5.6}$$

$$\text{Preis zu Ziel B: } P_B = c \cdot p_k \tag{5.7}$$

[12]Die durchaus übliche Anfahrtsgebühr wird hier vernachlässigt, da diese den gegebenen Sachverhalt nicht entscheidend ändert.

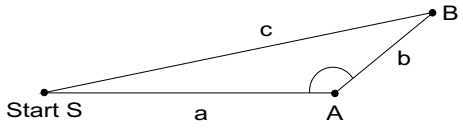


Abbildung 5.4: Modell für eine Taxifahrt mit zwei Wegpunkten.

Fahren beide gemeinsam, so liefert dies folgende Gesamtpreise, je nachdem, ob zuerst Punkt A oder Punkt B angefahren wird:

$$P_{AB} = a \cdot p_k + b \cdot p_k \tag{5.8}$$
$$P_{BA} = c \cdot p_k + b \cdot p_k \tag{5.9}$$

Die Variante besagt nun, dass der Preis P_{AB} geteilt wird. Damit jeder einen Nutzen hieraus zieht, muss für P_{AB} gelten:

$$\frac{1}{2} \cdot P_{AB} < P_A \quad \Leftrightarrow \quad \frac{a \cdot p_k + b \cdot p_k}{2} < a \cdot p_k \Rightarrow a > b \tag{5.10}$$
$$\frac{1}{2} \cdot P_{AB} < P_B \quad \Leftrightarrow \quad \frac{a \cdot p_k + b \cdot p_k}{2} < c \cdot p_k \Rightarrow c > \frac{a+b}{2} \tag{5.11}$$

Für P_{BA} gilt entsprechend:

$$\frac{1}{2} \cdot P_{BA} < P_B \quad \Leftrightarrow \quad \frac{c \cdot p_k + b \cdot p_k}{2} < c \cdot p_k \Rightarrow c > b \tag{5.12}$$
$$\frac{1}{2} \cdot P_{BA} < P_A \quad \Leftrightarrow \quad \frac{c \cdot p_k + b \cdot p_k}{2} < a \cdot p_k \Rightarrow a > \frac{b+c}{2} \tag{5.13}$$

Für die lokale Gruppenbildung von Punkt A ist nur die Profitmaximierung von Punkt A wesentlich. Ob die Route P_{AB} oder P_{BA} gewählt wird, hängt mit dem für A zu zahlenden Preis zusammen.

Damit die Route AB gewählt wird muss neben $a > b$ und $\frac{a+b}{2} < c$ gelten:

$$\frac{a+b}{2} < \frac{b+c}{2} \Rightarrow a < c$$

Für $c > b$, $c < a$ und $\frac{b+c}{2} < a$ wird die Route BA gewählt. Eine Profitmaximierung des Punktes B ist bei der lokalen Gruppierung nicht durchzuführen.

5.3.2.2 Variante 2: Anteilsmäßige Fahrtkosten

Variante 1 stellt die naheliegendste Preisbildungmöglichkeit dar, doch kommt es in dem Beispiel nur zur Gruppenbildung, wenn für die Strecken a und b aus Abbildung 5.4 $a > b$ gilt. Dies resultiert daraus, dass eine Person, die die Station A als Ziel hat, einen Teil der Reststrecke mitzahlen muss.

In Variante 2 werden die Teilstrecken anteilsmäßig verrechnet, d. h. bei gemeinsam gefahrenen Strecken wird der Fahrpreis geteilt.

Auch diese Variante soll an obigem Beispiel verdeutlicht werden. Ohne Gruppe berechnet sich der Fahrpreis gemäß den Gleichungen 5.6 und 5.7.

Bei Gruppenbildung würden sich folgende Preise ergeben:

$$\begin{aligned} P_{AB} &= a \cdot p_k + b \cdot p_k = P^G_{SA} + P^G_{AB} \\ P^G_{SA} &= \frac{a}{2} \cdot p_k \\ P^G_{AB} &= \frac{a}{2} \cdot p_k + b \cdot p_k \end{aligned}$$

Damit der Fahrpreis bei Gruppenbildung niedriger ist als bei einer Einzelfahrt, muss gelten:

$$\begin{aligned} P^G_{SA} = \tfrac{a}{2} \cdot p_k &< a \cdot p_k \Rightarrow \frac{a}{2} < a \quad (w) \\ P^G_{AB} = \tfrac{a}{2} \cdot p_k + b \cdot p_k &< c \cdot p_k \Rightarrow \frac{a}{2} + b < c \end{aligned}$$

In diesem Fall würde die Person mit Ziel A immer sofort eine Gruppe bilden wollen, da sie in allen möglichen Fällen weniger zahlen muss. Jedoch wäre noch zu testen, ob nicht der Weg über B nach A einen höheren Profit für A erbringen würde. Dies ist auf Grund geometrischer Gegebenheiten jedoch auszuschließen.

Um die mit einer zunehmenden Anzahl an Teilnehmern verbundene steigende Komplexität darzustellen, wird jetzt der Fall untersucht, in dem drei Personen eine Gruppe bilden, die, wie in Abbildung 5.5 zu sehen ist, an den Wegpunkten A, B und C ihr Ziel

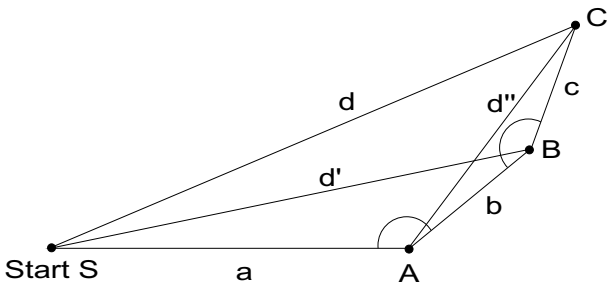


Abbildung 5.5: Modell für eine Taxifahrt mit drei Wegpunkten.

haben.

Würde jeder der drei Personen alleine fahren, entstünden folgende Preise:

$$\begin{aligned} P_A &= a \cdot p_k \\ P_B &= d' \cdot p_k \\ P_C &= d \cdot p_k \end{aligned}$$

Der Gesamtpreis bei Gruppenbildung beläuft sich auf

$$P_{ABC} = a \cdot p_k + b \cdot p_k + c \cdot p_k = P^C_A + P^C_B + P^C_B.$$

Wieder werden die Kosten für gemeinsam gefahrene Strecken aufgeteilt:

$$\begin{aligned} &\text{Preis zu Wegpunkt A:} && \tfrac{a}{3} \cdot p_k &< a \cdot p_k \Rightarrow \frac{a}{3} < a \quad (w), \\ &\text{Preis zu Wegpunkt B:} && \tfrac{a}{3} \cdot p_k + \tfrac{b}{2} \cdot p_k &< d' \cdot p_k \Rightarrow \frac{a}{3} + \frac{b}{2} < d', \\ &\text{Preis zu Wegpunkt C:} && \tfrac{a}{3} \cdot p_k + \tfrac{b}{2} \cdot p_k + c \cdot p_k &< d \cdot p_k \Rightarrow \frac{a}{3} + \frac{b}{2} + c < d. \end{aligned}$$

Dies sind jedoch nicht alle Bedingungen. Es könnten sich Personen entschließen nur eine Zweiergruppe zu bilden, wenn sie so günstiger befördert würden. Im vorliegenden Fall könnten nur die Wegpunkte A und C oder nur die Wegpunkte B und C angefahren werden. Dies fügt zwei weitere Nebenbedingungen hinzu:

$$\begin{aligned}
\text{Nebenbedingung 1:} \quad & \tfrac{a}{3} \cdot p_k + \tfrac{b}{2} \cdot p_k + c \cdot p_k < \frac{a}{2} \cdot p_k + d'' \cdot p_k \Rightarrow, \\
\text{Nebenbedingung 2:} \quad & \tfrac{a}{3} \cdot p_k + \tfrac{b}{2} \cdot p_k + c \cdot p_k < \frac{d'}{2} \cdot p_k + c \cdot p_k \Rightarrow \frac{a}{3} + \frac{b}{2} < \frac{d'}{2}.
\end{aligned}$$

Diese fünf Bedingungen können zu drei zusammengefasst werden, die alle erfüllt sein müssen, damit eine Gruppe mit drei Teilnehmern zu Stande kommt:

$$\begin{aligned}
d &> \frac{a}{3} + \frac{b}{2} + c, \\
d' &> \frac{2}{3}a + b, \\
d'' &> \frac{a}{6} + \frac{b}{2} + c.
\end{aligned}$$

5.3.2.3 Variante 3: Gleiche prozentuale Ersparnis

In der vorigen Variante 2 profitierte die Person, die im Wegpunkt A das Taxi verlässt am meisten, da bei einer Gruppengröße von n Personen der Fahrpreis für das erste Teilstück auf n Personen verteilt wird.

In der letzten hier vorgestellten Variante soll jeder Mitfahrer den gleichen prozentualen Nutzen durch die Gruppe bekommen, d. h. wenn eine Person als Gruppenmitglied nur 70 % des Fahrpreises zahlt, dann auch alle anderen.

Ohne Gruppenbildung entspricht der Fahrpreis den Gleichungen 5.6 und 5.7. Mit Gruppe ergibt sich:

$$\begin{aligned}
P_{AB} &= a \cdot p_k + b \cdot p_k = P_A^C + P_B^C, \\
P_A^C &= a \cdot p_k \cdot q, \\
P_B^C &= c \cdot p_k \cdot q.
\end{aligned}$$

Hierbei ist q der Ersparnisquotient, d. h. um auch eine Ersparnis zu erreichen, muss stets $q < 1$ gelten. q lässt sich angeben:

$$q = \frac{a+b}{a+c}, \qquad q < 1$$

Bei dieser Variante kommt es zur Gruppenbildung, wenn q kleiner als eins ist. Dies bedeutet

$$\frac{a+b}{a+c} < 1 \Rightarrow a + b < a + c \Rightarrow b < c,$$

d. h. für eine Gruppenbildung muss in diesem Fall der Weg von Punkt A nach Punkt B kleiner sein als vom Startpunkt zu Punkt B, weil im anderen Fall der direkte Weg für Punkt B günstiger ist.

Doch es muss noch die Route BA überprüft werden. Hierbei ergibt sich mit

$$\begin{aligned} P_{BA} &= c \cdot p_k + b \cdot p_k = P^G_{SA} + P^G_{SC}, \\ P^G_{SA} &= a \cdot p_k \cdot q, \\ P^G_{SC} &= c \cdot p_k \cdot q, \end{aligned}$$

ein Ersparnisquotient von

$$q = \frac{c+b}{a+c}, \qquad q < 1,$$

d. h. eine Gruppebildung kommt im Falle $b < a$ zu Stande.

Wann welche Route gewählt wird, ist aus

$$\frac{a+b}{a+c} < \frac{c+b}{a+c} \Rightarrow a < c$$

ersichtlich.

5.3.2.4 Vergleich der Preisbildungsvarianten

In diesem Abschnitt sollen die drei präsentierten Varianten verglichen werden. Dies soll zuerst an einem konkreten Beispiel erfolgen. Danach werden die drei Variante analytisch miteinander verglichen.

Im gewählten Beispiel sollen zwei Wegpunkte A und B vorhanden sein und ferner sei a=3 km, b=4 km und c=6 km. Der Preis pro Kilometer sei 1,00 €. In Tabelle 5.1 sind die Kosten je nach Variante aufgetragen. Zu bemerken ist, dass die Person mit Ziel Wegpunkt

	ohne Gruppe	Variante 1	Variante 2	Variante 3
Kosten zu WP A	3,00 €	k.G.	1,50 €	2,33 €
Kosten zu WP B	6,00 €	3,50 €	5,50 €	4,67 €

Tabelle 5.1: Beispiel für die drei Preisvarianten mit zwei Wegpunkten

A und Variante 1 keine Gruppe (k.G. in Tabelle 5.1) in bilden, da die Kosten 3,50 € betragen, ohne Gruppe jedoch nur 3,00 €. Die andere Person würde jedoch eine Gruppe bilden wollen, da sie 2,50 € weniger zahlen müsste. In Variante 3 ist die Ersparnis für beide jeweils ungefähr 22 % vom Preis ohne Gruppenbildung.

Die Gruppierungsbedingung kann für dieses Beispiel konkret angegeben werden. Da die Fahrgäste immer günstiger fahren wollen, muss für $x \in I_A = [1..3[$ und für $y \in I_B = [4..6[$ die Nebenbedingung $x + y = 7$ gelten.

Dies führt unweigerlich zu einer weiteren, vierten Variante, welche aus den Intervallen I_A bzw. I_B willkürlich eine Zahl zieht und die andere errechnet. Im Mittel würden die Kosten zu A 2 € und zu B 5 € betragen.

Um eine allgemeine Aussage treffen zu können, welche Variante in den meisten Fällen zur Gruppenbildung führt, wird der in Abbildung 5.4 eingetragene Winkel δ für die drei Varianten betrachtet. Diese sind in Tabelle 5.2 angegeben.

Der Bereich des Winkel δ ist von zwei Variablen abhängig, die jedoch durch $x = \frac{a}{b}$ substituiert werden können. Somit kann für jedes x die Wahrscheinlichkeit für eine Gruppenbildung berechnet werden. Die entsprechenden Funktionen sind in Tabelle 5.3 dargestellt.

Variante 1	$\cos\delta \leq \frac{3}{8}(\frac{a}{b} + \frac{b}{a}) - \frac{1}{4}$
Variante 2	$\cos\delta \leq \frac{3}{8}\frac{a}{b} - \frac{1}{2}$
Variante 3	$\cos\delta \leq \frac{1}{2}\frac{a}{b}$

Tabelle 5.2: Winkelbeziehung bei den Preisvarianten

Variante 1	$p_{V_1} = \begin{cases} 0 & x < 1, \\ 1 - \frac{1}{\pi}\arccos(\frac{3}{8}(x + \frac{1}{x}) - \frac{1}{4}) & x < 3, \\ 1 & \text{sonst.} \end{cases}$
Variante 2	$p_{V_2} = \begin{cases} 0 & x < 0, \\ 1 - \frac{1}{\pi}\arccos(\frac{3}{8}x - \frac{1}{2}) & x < 4, \\ 1 & \text{sonst.} \end{cases}$
Variante 3	$p_{V_3} = \begin{cases} 0 & x < 0, \\ 1 - \frac{1}{\pi}\arccos(\frac{1}{2}x) & x < 2, \\ 1 & \text{sonst.} \end{cases}$

Tabelle 5.3: Wahrscheinlichkeiten für die einzelnen Preisvarianten

Zu erkennen ist, dass für $x > 4$ stets eine Gruppenbildung zu Stande kommt. Um zu eruieren, welche Variante am häufigsten Gruppen bildet, muss somit nur das Intervall $[0; 4]$ betrachtet werden. Für jede Variante ist der Funktionsverlauf innerhalb dieses Intervalls in Abbildung 5.6 skizziert.

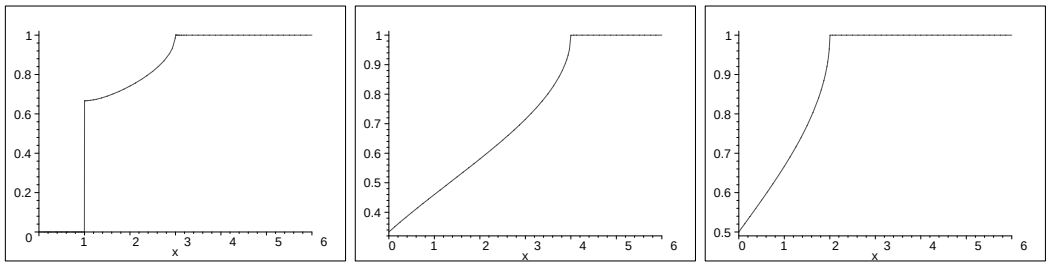


Abbildung 5.6: Visualisierung der Wahrscheinlichkeiten der einzelnen Varianten. Gleiche Fahrtkosten a), anteilsmäßige Fahrtkosten b), gleiche prozentuale Ersparnis c).

Eine Integration der einzelnen Wahrscheinlichkeiten p_{V_x} entlang des Intervalls dient

der Ableitung von Rückschlüssen über die Häufigkeit der Gruppenbildungen.

$$\int_0^4 p_{V_1}\,dx = \quad \tfrac{26}{9} - \tfrac{6\sqrt{3}}{9\pi} \approx 2,521 \tag{5.14}$$

$$\int_0^4 p_{V_2}\,dx = \quad \tfrac{28}{9} - \tfrac{12\sqrt{3}}{9\pi} \approx 2,376 \tag{5.15}$$

$$\int_0^4 p_{V_3}\,dx = \quad 4 - \tfrac{2}{\pi} \approx 3,363 \tag{5.16}$$

Wie aus den Gleichungen 5.14-5.16 ersichtlich ist, wird bei Variante 3 am häufigsten eine Gruppe gebildet. Variante 2 schneidet am schlechtesten ab. Da bei dieser Variante der erste Teilnehmer sehr stark von der Gruppenbildung profitiert, sind nur schwer weitere Gruppenmitglieder zu finden, die ebenfalls profitieren.

Als weiterführende Verfahren können die Fahrpreise verhandelt werden. Hierbei müssten zuerst adäquate Verhandlungsprotokolle entwickelt werden, die eine Verhandlung in einem ad hoc Umfeld möglich machen. Ziel ist es, möglichst schnell und mit wenig Kommunikationsaufwand eine Verhandlung zu führen.

5.4 Gruppierung im Taxi-Sharing-Szenario

Für die Taxi-Sharing Anwendung werden sämtliche Algorithmen aus Abschnitt 4.3 implementiert. Der Fokus dieses Abschnitts liegt jedoch in der Durchführung der Gruppierung.

Bezüglich der anderen, in Abschnitt 4.3 beschriebenen Algorithmen, sei erwähnt, dass bei der Initiatorbestimmung die Methode des niedrigsten Identifikators verwendet wird.

Die Segmentierung ist so realisiert, dass der Designparameter k - der die Größe des Segments bestimmt - nicht festgelegt ist, sondern sich ändern lässt. Auf diese Weise kann der zweckmäßigste Wert für k eruiert werden.

Hinsichtlich der virtuellen Topologien wird das Taxi-Sharing-Szenario sowohl mit der Baum- als auch der Ringstruktur getestet. Grund hierfür ist, dass untersucht wird, welche Topologie sich besser für das Szenario eignet.

5.4.1 Lokale Gruppenbildung

Um eine Lokale Gruppe zu erhalten, sind in Abschnitt 4.3.4.2 mehrere Heuristiken beschrieben. Was nicht explizit erwähnt wird, aber aus Gleichung 5.4 hervorgeht, ist, dass der lokale Gruppierungsvorgang auch von der verwendeten Preisvariante abhängt. Dies hat Auswirkungen auf die Verwendung der Heuristik.

Im Abschnitt 5.3.2 existieren drei verschiedene Preisvarianten. Sollen die Heuristiken verwendet werden, müsste für jede Preisvariante eine eigene Heuristik entwickelt werden.

Aus diesem Grund fiel der Entschluss, das lokale Gruppieren mit der Lokales-Maximum-Heuristik durchzuführen. Implementiert ist diese gemäß dem Algorithmus in Abbildung 4.31. Zur Veranschaulichung ist das Verfahren nochmals in Abbildung 5.7 wiedergegeben. In Abbildung 5.7a bis 5.7f wird sukzessive immer das Gruppierungsobjekt zur lokalen Gruppe hinzugefügt, welches $\pi(\mathcal{G})$ maximiert. Im Fall des Taxi-Sharing entspricht ein neues Gruppenmitglied stets dem nächstgelegenen Objekt hinsichtlich der Euklidischen

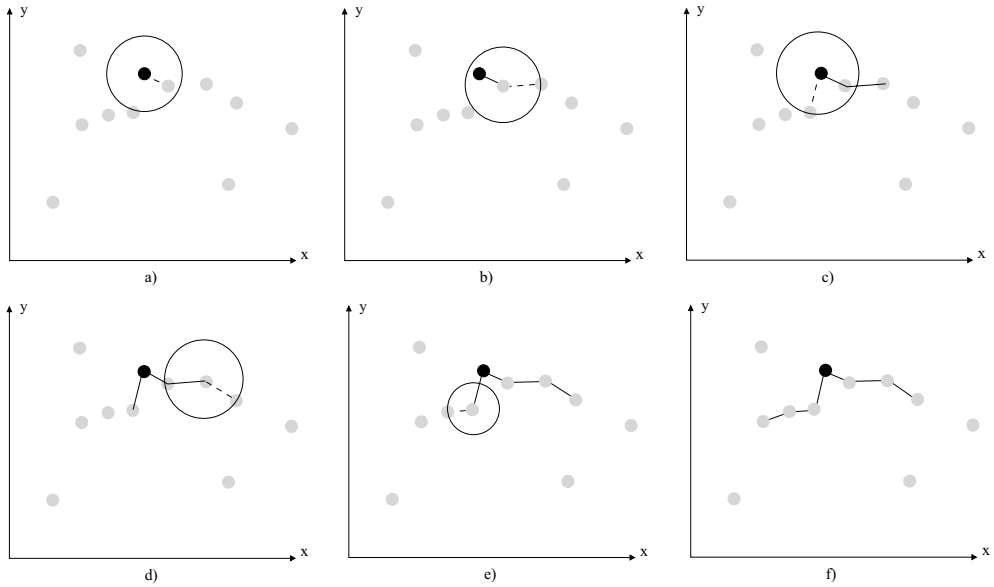


Abbildung 5.7: Lokales-Maximum-Heuristik beim Taxi-Sharing

Distanz. Das Hinzufügen endet, wenn hierdurch der Nutzen nicht weiter gesteigert werden kann.

5.4.2 Verteilte Gruppenbildung

Für die verteilte Gruppenbildung ist die Angabe von Bedingungen notwendig (siehe Gleichungen 4.30-4.32), um den Algorithmus in Abschnitt 4.3.4.3 verwirklichen zu können. Diese Bedingungen hängen jedoch von der jeweiligen Preisstrategie ab.

Abbildung 5.8 zeigt die hierfür verwendete Bezeichnung für Änderungen einer Gruppe.

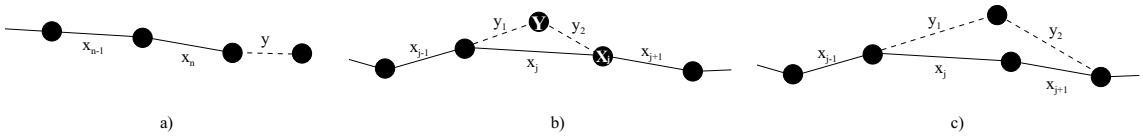


Abbildung 5.8: Einfügen und Ersetzen eines Punktes im Profilnetzwerk

Abbildung 5.8a zeigt das Anfügen eines Objektes. Das Gewicht des neu hinzuzufügenden Objektes wird mit y bezeichnet und entspricht dem Fahrpreis zwischen den Knoten. Beim Einfügen in Abbildung 5.8b wird das Gewicht x_j durch die Gewichte y_1 und y_2 ersetzt. Dieses Bild dient auch zur Veranschaulichung des Löschens eines Objekts. In diesem Fall wird y_1 und y_2 durch x_j ersetzt. Abbildung 5.8c stellt das Ersetzen eines Objektes dar. Hier wird x_j und x_{j+1} durch y_1 und y_2 ersetzt.

In Abbildung 5.8 werden die Gewichte mit Kleinbuchstaben, wie z. B. x_1 oder y bezeichnet; Gruppierungsobjekte hingegen mit Großbuchstaben, z. B. Y oder X_j.

Nachfolgend werden für alle in Abschnitt 5.3.2 vorgestellten Preisvarianten Bedingungen generiert, mit denen es möglich ist zu entscheiden, ob in einer Gruppe ein Teilnehmer hinzugefügt, gelöscht oder ersetzt werden kann.

5.4.2.1 Gleiche Fahrtkosten

Für die erste Variante ist für Gruppen mit n bzw. $n+1$ Teilnehmern die Preisfunktion $P_{\mathcal{G}}^n$ und $P_{\mathcal{G}}^{n+1}$ durch

$$\begin{aligned} P_{\mathcal{G}}^n &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ P_{\mathcal{G}}^{n+1} &= \frac{1}{n+1}\Big(\sum_{i=1}^{n} x_i + y\Big) \end{aligned}$$

gegeben. Damit ein Anfügen von neuen Gruppenmitgliedern erfolgt, muss

$$P_{\mathcal{G}}^{n+1} < P_{\mathcal{G}}^n \quad \Leftrightarrow \quad P_{\mathcal{G}}^n > y \tag{5.17}$$

erfüllt sein, d. h. das Gewicht zum neuen Element muss größer sein als die Profitfunktion.

Das Anfügen von Gruppierungsobjekten ist ein Sonderfall des Einfügens eines Gruppierungsobjektes. Für die Verallgemeinerung gilt hinsichtlich der Nutzenfunktion π:

$$\begin{aligned} P_{\mathcal{G}}^n &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ P_{\mathcal{G}}^{n+1} &= \frac{1}{n+1}\Big(\sum_{i=1}^{n} x_i - x_j + y_1 + y_2\Big). \end{aligned}$$

Hieraus resultiert als die Bedingung für das Einfügen:

$$P_{\mathcal{G}}^{n+1} < P_{\mathcal{G}}^n \quad \Leftrightarrow \quad P_{\mathcal{G}}^n > y_1 + y_2 - x_j. \tag{5.18}$$

Aus Gleichung 5.18 folgt unmittelbar auch die Bedingung für das Löschen eines Gruppierungsobjektes aus der Gruppe, mit

$$P_{\mathcal{G}}^n < P_{\mathcal{G}}^{n+1} \quad \Leftrightarrow \quad P_{\mathcal{G}}^n < y_1 + y_2 - x_j. \tag{5.19}$$

Liegt eine Gruppe vor, kann mit Gleichung 5.19 überprüft werden, ob der Nutzen durch Entfernen eines Gruppenteilnehmers erhöht werden kann.

Um das Löschen und gleichzeitige Hinzufügen eines Punktes zu ermöglichen wird mit

$$y_1 + y_2 > x_j + x_{j+1} \tag{5.20}$$

noch die Bedingung für das Ersetzen eines Gruppenmitglieds angegeben. Somit muss die Summe zweier neuer Gewichte kleiner sein als die zu ersetzenden.

Schließlich kann es noch geschehen, dass das Anfügen eines einzelnen Gruppierungsobjektes zur Gruppe zunächst keinen zusätzlichen Nutzen erbringt. Durch das Anfügen mehrerer Gruppierungsobjekte würde sich hingegen der Nutzen erhöhen. Dieser Fall wurde von keinem der bisherigen Bedingungen abgedeckt. Die Funktion π nimmt durch Hinzukommen von zusätzlichen j Teilnehmern den Wert

$$P_{\mathcal{G}}^{n+j} = \frac{1}{n+j}\Big(\sum_{i=1}^{n+j} x_i\Big)$$

an. Für einen größeren Profit muss schließlich

$$P_{\mathcal{G}}^{n+j} < P_{\mathcal{G}}^{n} \quad \Leftrightarrow \quad P_{\mathcal{G}}^{n} > \frac{1}{j}\sum_{i=n+1}^{n+j} \tag{5.21}$$

gelten.

5.4.2.2 Anteilsmäßige Fahrtkosten

Bei dieser Variante ist die Angabe einer Bedingung für das Anfügen nicht notwendig. Jedes bisherige Gruppenmitglied wird durch Anfügen eines neuen Mitgliedes einen höheren Profit erzielen und somit weniger für die Taxifahrt zahlen müssen.

Soll ein zusätzliches Gruppierungsmitglied an der Stelle j eingefügt werden, so würde dies für sämtliche Gruppenmitglieder, deren Reiseziel vor dem von j stehen, den Nutzen erhöhen, für welche jedoch $i > j$ gilt, würde der Nutzen sinken.

Die einzige Möglichkeit den Nutzen zu erhöhen ist somit einzelne Gruppierungsteilnehmer auszutauschen. Hierzu muss

$$\frac{y_1}{n-j+1} + \frac{y_2}{n-j} < \frac{x_j}{n-j+1} + \frac{x_{j+1}}{n-j} \tag{5.22}$$

erfüllt sein.

5.4.2.3 Gleiche prozentuale Ersparnis

Damit bei der dritten Variante der Nutzen ansteigt, muss der Ersparnisquotient q durch Hinzunehmen neuer Mitglieder abnehmen. Soll ein Objekt hinzugefügt werden, entspricht dies folgenden Werten:

$$\begin{aligned} q_n &= \frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} \pi_0(x_i)}, \\ q_{n+1} &= \frac{\sum_{i=1}^{n} x_i + y}{\sum_{i=1}^{n} \pi_0(X_i) + \pi_0(Y)}. \end{aligned}$$

Hierbei ist $\pi_0(x_i)$ der Wert, den das Objekt X_i (siehe Abbildung 5.8b) ohne Gruppe besitzt.

Damit q abnimmt, muss

$$q_{n+1} < q_n \quad \Leftrightarrow \quad q_n > \frac{y}{\pi_0(Y)} \tag{5.23}$$

gelten.

Beim Einfügen eines Gruppierungsobjekts gilt für q_n und q_{n+1} :

$$\begin{aligned} q_n &= \frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} \pi_0(X_i)}, \\ q_{n+1} &= \frac{\sum_{i=1}^{n} x_i - x_j + y_1 + y_2}{\sum_{i=1}^{n} \pi_0(X_i) + \pi_0(Y_1)}, \qquad \big(\pi_0(X_j) = \pi_0(Y_2)\big). \end{aligned}$$

Hieraus folgt als Bedingung

$$q_{n+1} < q_n \quad \Leftrightarrow \quad q_n > \frac{x_j + y_1 + y_2}{\pi_0(Y_1)}. \tag{5.24}$$

Wie bereits bei Variante 1 kann auch hier eine Bedingung für das Löschen eines Gruppenelements bestimmt werden. Auf die explizite Darstellung des Ergebnisses wird hier verzichtet, weil es der inversen Anforderung des Hinzufügens eines neuen Teilnehmers entspricht.

Ein Ersetzen eines Gruppenmitglieds durch ein anderes kann durchgeführt werden, wenn

$$q_n > \frac{x_j + x_{j+1} + y_1 + y_2}{\pi_0(X_j) + \pi_0(X_{j+1}) + \pi_0(Y_1) + \pi_0(Y_2)} \tag{5.25}$$

erfüllt ist.

Schließlich ist auch hier noch der Fall abzudecken, wenn nur das Hinzufügen von mehreren Teilnehmern zu einer Profitzunahme führt. Der Ersparnisquotient q definiert sich bei Hinzunahme von j Teilnehmern durch

$$q_{n+j} = \frac{\sum_{i=1}^{n+j} x_i}{\sum_{i=1}^{n+j} \pi_0(X_i)}.$$

Diese j Teilnehmer können hinzugefügt werden, wenn

$$q_{n+j} < q_n \quad \Leftrightarrow \quad q_n > \frac{\sum_{i=n+1}^{n+j} x_i}{\sum_{i=n+1}^{n+j} \pi(X_i)} \tag{5.26}$$

erfüllt ist.

5.4.3 Heuristiken

Nachdem mögliche Heuristiken bereits in Abschnitt 4.3.4.2 präsentiert wurden, sollen diese anhand des Taxi-Sharing Szenarios in diesem Abschnitt konkretisiert werden.

5.4.3.1 Gebietsdefinition

Durch die Definition eines Gebietes innerhalb des Profilsraumes soll die Anzahl an zu testenden Profilpunkten reduziert werden, wie bereits in Abschnitt 4.3.4.2 erläutert wurde. Dargestellt ist dies in Abbildung 5.9. In Abbildung 5.9a ist der Startpunkt S und ein erster Haltepunkt Z zu sehen. Ferner sind mehrere Punkte zu erkennen, die die Ziele der

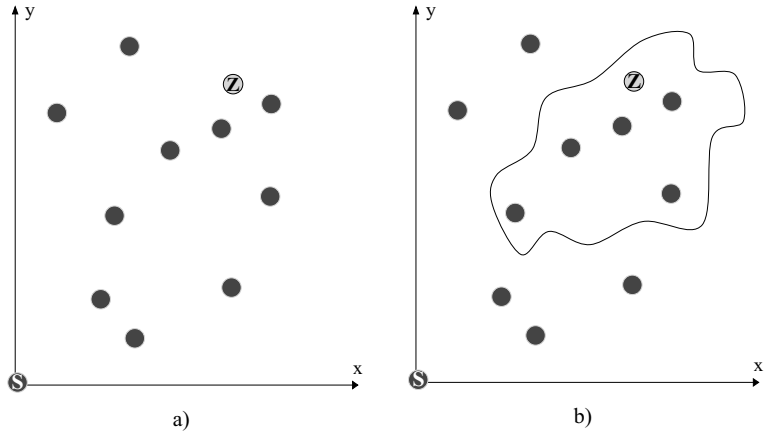


Abbildung 5.9: Reduzierung der Profilpunkte im Profilnetzwerk durch ein einschränkendes Gebiet

potenziellen Mitfahrer symbolisieren. Durch das Gebiet in Abbildung 5.9b wird diese Zahl erheblich reduziert.

Der zu definierende Bereich ist jedoch abhängig von der verwendeten Preisstrategie, d. h. von den drei in Abschnitt 5.3.2 präsentierten Varianten, weshalb nachfolgend für alle Varianten das Gebiet bestimmt wird.

Gleiche Fahrtkosten: Sei a die Entfernung von S zum ersten Haltepunkt Z, dann liefern die Gleichungen 5.13 folgendes Gebiet:

$$\left(\frac{(x-\frac{1}{2}a)}{a}\right)^2+\left(\frac{y}{\frac{3}{4}a}\right)^2=1 \tag{5.27}$$

Gleichung 5.27 definiert somit eine Ellipse. Da jedoch für die Gruppenbildung die erste Strecke länger als die zweite sein muss, reduziert sich das Gebiet auf die rechte Hälfte der Ellipse, wie in Abbildung 5.10a ersichtlich ist.

Anteilsmäßige Fahrtkosten: Variante 2 bildet eine Ausnahme bei der Gebietsbestimmung. Da der Fahrpreis sich mit jedem hinzukommenden Fahrgast verringert, werden Mitfahrer, die ein späteres Ziel haben, generell hinzugefügt. Andererseits existieren keine Fahrgäste, die zwischen S und Z das Taxi verlassen und gleichzeitig den Profit von Z erhöhen. Das Gebiet, welches zur Gruppierung herangezogen werden kann, besteht somit aus der kompletten Ebene. Für Variante 2 kann folglich kein einschränkendes Gebiet angegeben werden.

Gleiche prozentuale Ersparnis: Aus der Bedingung $b < a$ resultiert für diese Preisbildungsvariante ein Kreis um den Punkt Z mit dem Radius a. Sämtliche Punkte innerhalb dieses Kreises würden für die Gruppenbildung in Frage kommen. In Abbildung 5.10b ist dies verdeutlicht.

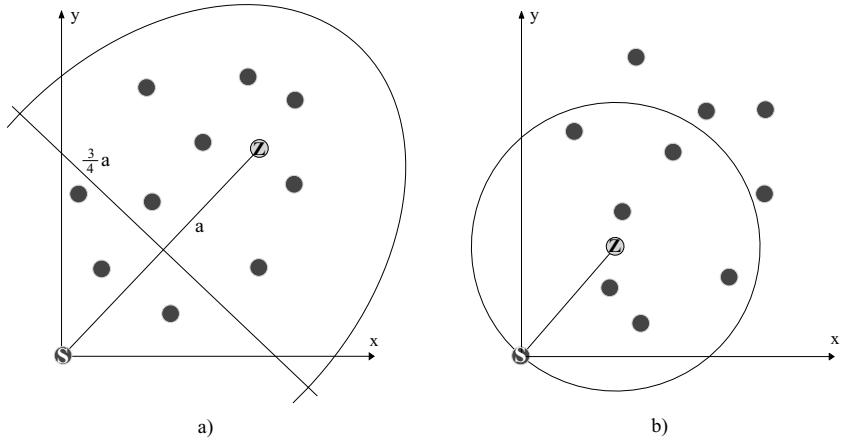


Abbildung 5.10: Gebiete für die einzelnen Preisvarianten

5.4.3.2 Lokale-Minimum-Heuristik

Wie in Abschnitt 4.3.4.2 bereits abstrakt erläutert, wird bei der lokalen-Minimum-Heuristik stets der im Profilraum nächstgelegene Punkt gewählt, der die Payoff-Funktion maximal vergrößert.

Für das Taxi-Sharing-Szenario bedeutet dies, dass der nächste Profilpunkt das jeweils nächstgewählte Gruppierungsobjekt darstellt. Veranschaulicht wird dies in Abbildung 5.11. Ausgehend von einem Startpunkt, wird sukzessive stets der Punkt mit der kürzesten Entfernung als nächster Gruppenteilnehmer gewählt, wie in Abbildung 5.11 ersichtlich ist. Dies wird solange wiederholt, wie die notwendigen und hinreichenden Bedingungen gemäß den Gleichungen 4.9 und 4.10 erfüllt sind.

5.5 Domänenbeschreibung für das Taxi-Sharing Szenario

In Abschnitt 4.4 wurden die Meta-Beschreibungen für Profil- und Gruppendefinition eingeführt. In diesem Abschnitt wird die Profil- und Gruppenbeschreibung als XML-Schema für das Taxi-Sharing-Szenario präsentiert. Anschließend wird für beide Beschreibungen eine Instanz in XML angegeben, um zu verdeutlichen, wie ein verwendbares Profil auf dem Endgerät aussehen kann.

5.5.1 Profildefinition für das Taxi-Sharing Szenario

Gemäß Abschnitt 4.4.1.1 besteht eine Profildefinition aus den Benutzerdaten (`<USERDATA>`) und aus einer Sequenz von `<PROFILEENTRY>`-Tags, die aus einem `<ENTRYNAME>` und einem `<ENTRYVALUE>` bestehen. Im Folgenden wird spezifiziert, welche `<PROFILEENTRY>`-Tags für das Taxi-Sharing-Szenario nötig sind.

Als Profileinträge sind in Abbildung 5.12 drei `<PROFILEENTRIES>` definiert. Die Einträge `xValue` und `yValue` definieren den Zielpunkt, der in der konkreten Anwendung des

Abbildung 5.11: Lokale-Minimum-Heuristik im Taxi-Sharing Szenario: Es wird jeweils der nächstgelegene Profilpunkt gewählt.

Taxi-Sharings einem Punkt in einem zweidimensionalen Koordinatensystem entspricht. Der letzte Eintrag `maxTime` definiert eine zeitliche Oberschranke für die Dauer einer Fahrt.

5.5.2 Gruppendefinition

Nach der Profilbeschreibung wird in diesem Abschnitt die Gruppenbeschreibung für das Taxi-Sharing präsentiert.

In Abbildung 5.13 ist eine Gruppendefinition für das Taxi-Sharing zu sehen. Zuerst wird mit dem Eintrag `profit` im `<GROUPTYPE>` Tag definiert, dass es sich um nutzenbasiertes Gruppieren handelt. Die zu gruppierenden Profilteile sind die `xValue` und `yValue` Einträge, welche der Zielkoordinate entsprechen. Beide Elemente müssen Teil des Profils

```
<?xml version="1.0" standalone="yes" ?>
<USERPROFILE>
    <USERDATA>
        <USERID>123456789abc</USERID>
        <NAME>Max Mustermann</NAME>
    </USERDATA>
    <PROFILEENTRIES>
        <PROFILEENTRY>
            <ENTRYNAME>xValue</ENTRYNAME>
            <ENTRYNUMBERVALUE>190</ENTRYNUMBERVALUE>
        </PROFILEENTRY>
        <PROFILEENTRY>
            <ENTRYNAME>yValue</ENTRYNAME>
            <ENTRYNUMBERVALUE>251</ENTRYNUMBERVALUE>
        </PROFILEENTRY>
        <PROFILEENTRY>
            <ENTRYNAME>maxTime</ENTRYNAME>
            <ENTRYNUMBERVALUE>25</ENTRYNUMBERVALUE>
        </PROFILEENTRY>
    </PROFILEENTRIES>
</USERPROFILE>
```

Abbildung 5.12: Beispiel einer Profilbeschreibung

```
<?xml version="1.0" standalone="yes" ?>
<GROUPDESCRIPTION>
    <GROUPTYPE>profit</GROUPTYPE>
    <ENTRIES>
        <ENTRYNAME>xValue</ENTRYNAME>
        <ENTRYNAME>yValue</ENTRYNAME>
    </ENTRIES>
    <GROUPSIZE>
        <LOWERLIMIT>1</LOWERLIMIT>
        <UPPERLIMIT>4</UPPERLIMIT>
    </GROUPSIZE>
    <VALIDITYPERIOD>
        <START>2005-07-21T00:00:00</START>
        <END>2005-07-22T00:00:00</END>
    <VALIDITYPERIOD>
</GROUPDESCRIPTION>
```

Abbildung 5.13: Beispiel für eine Gruppenbeschreibung

sein, was, wie in Abbildung 5.12 zu sehen, erfüllt ist. Der dritte Wert `maxTime` aus Abbildung 5.12 ist zwar ein Profileintrag, wird aber für das Gruppieren nicht herangezogen.

Als letzte Gruppeneigenschaft wird die Gruppengröße angegeben. Für das Taxi-Sharing-Szenario ist eine adäquate Gruppengröße das Intervall $[1; 4]$, d. h. im ungünstigsten Fall muss eine Person alleine fahren und vier Personen ist die Obergrenze, denn damit ist ein gewöhnliches Taxi an der Grenze seiner Kapazität, die für die Mitfahrer als angenehm empfunden wird.

Als Letztes ist noch die zeitliche Gültigkeit der Gruppe angegeben. Die Gruppendefinition ist somit am 21. Juli 2005 den ganzen Tag gültig.

5.5.3 Domänendefinition

In der Domänendefinition werden domänenabhängige Elemente durch Interfaces beschrieben sowie das MoPiDiG Framework konfiguriert. Es kann festgelegt werden, ob eine virtuelle Topologie verwendet werden soll und im Falle einer positiven Entscheidung kann zwischen der Baum- und Ringtopologie ausgewählt werden.

Im Falle des Taxi-Sharing Szenarios müssen die Nutzenfunktion und die Gruppierungsbedingungen in der Domänendefinition festgelegt werden. Dies wird durch Implementieren vorgegebener Schnittstellen (Interfaces) erreicht.

5.6 Simulationsumgebung

Für einen Test der Gruppenbildung in einer realen Umgebung würden mehrere Anwender und somit auch mobile Endgeräte benötigt. Da eine Gruppe mindestens aus zwei Teilnehmern bestehen muss, ist diese Anzahl als Minimum anzusehen - jedoch nicht ausreichend. Um adäquate Ergebnisse zu erhalten wären fünf bis zehn Anwender notwendig. Ferner wären Testpersonen erforderlich, die jedoch nicht zur Verfügung standen.

Bei der Beantwortung von Skalierbarkeitsfragen würden noch zusätzliche Geräte notwendig werden, deren Anzahl a priori nicht feststeht.

Diese Überlegungen führten dazu, dass bevor ein Test in realer Umgebung durchgeführt wird, mehrere Simulationen durchgeführt werden, um auf diese Weise das Verhalten der Gruppenbildung überprüfen zu können.

In diesem Abschnitt wird die entwickelte Simulationsumgebung vorgestellt. Diese besteht aus einem visualisierten konfigurierbaren Bewegungsmodell und einer Simulation der Gruppenerstellung.

5.6.1 Bewegungsmodelle

Abbildung 5.14 zeigt die entwickelte Simulationsumgebung. Sie erlaubt es folgende Parameter zu verändern:

- Geschwindigkeit v der Teilnehmer,
- Knotendichte ρ (Fläche konstant, Knotenanzahl variiert), d. h. Endgeräte/m^2,
- Kommunikationsradius r_K der Endgeräte,
- Bewegungsrichtung (willkürlich vs. Vorzugsrichtung) der Teilnehmer,
- Simulationsbereich (offener vs. begrenzter) und die
- Anzahl an Hops.

Die große Fläche im Simulationsfenster stellt den Ort dar, in dem die Gruppierung stattfindet. Dies kann z. B. ein Bahnhofsgebäude, die Gepäckausgabe an Flughäfen oder ein sonstiges stark frequentiertes Areal sein.

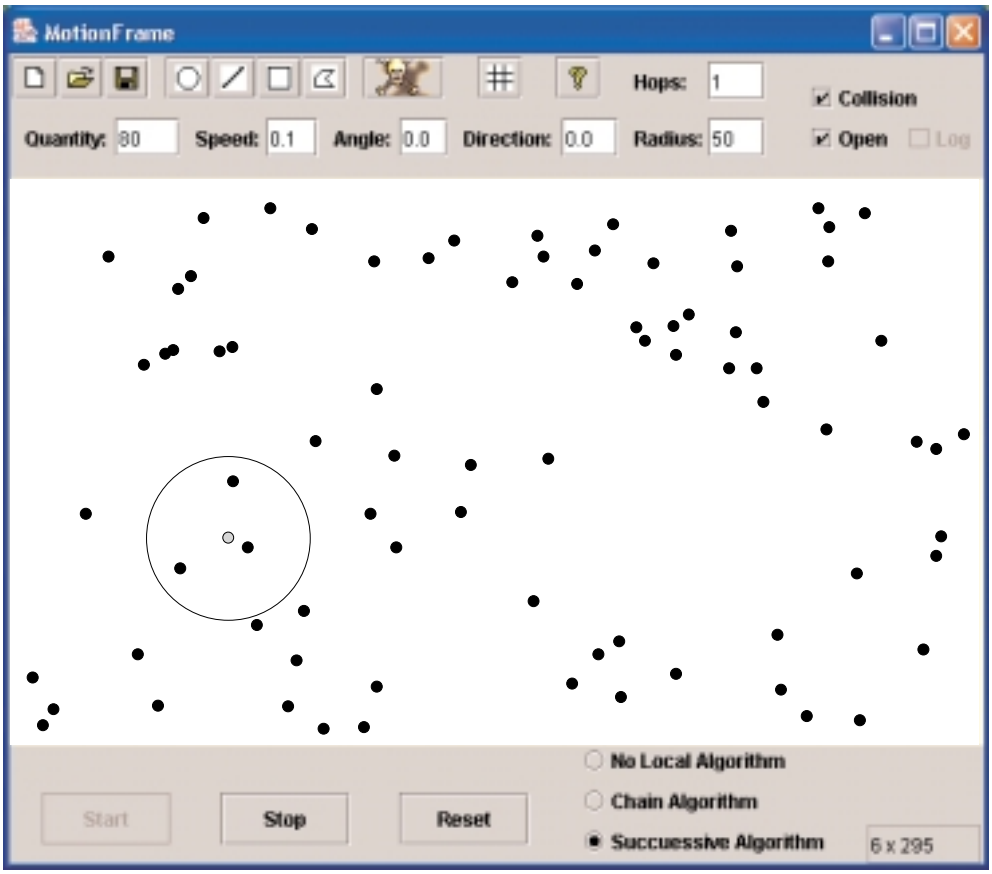


Abbildung 5.14: Simulationsumgebung für die Nachbildung der Bewegung der mobilen Endgeräte

Die Punkte, die in Abbildung 5.14 zu sehen sind, spiegeln mobile Endgeräte wider. Die Anzahl der Punkte kann vor Beginn einer Simulation durch das Feld „Quantity“ festgelegt werden.

Mit jedem Punkt ist ein Geschwindigkeitsvektor $\vec{v}$ assoziiert, welcher die Geschwindigkeit und die Richtung des Punktes beinhaltet. Dieser Geschwindigkeitsvektor wird zu Beginn einer Simulation willkürlich gewählt und bleibt auch während der kompletten Simulationsdauer unverändert, sofern nicht ein Hindernis oder eine Grenzfläche den Weg versperrt. In solch einem Fall bleibt der Betrag des Vektors unverändert, nur die Richtung ändert sich gemäß Einfallswinkel gleich Ausfallswinkel. Ob auch andere Punkte als Hindernisse anzusehen sind, kann durch das Kontollkästchen „Collision“ festgelegt werden.

In der Simulationsumgebung wird im „Speed“ Eingabefeld ein Mittelwert $\vec{v}$ spezifiziert. Die Auswahl der Geschwindigkeit erfolgt aus dem Intervall $[0; 2 \cdot \bar{v}]$, wobei die Gewichtung gleichverteilt erfolgt.

Bezüglich der Spezifikation der Richtung der Punkte sind zwei Parameter notwendig, um jeweils die Vorzugsrichtung entlang der x- und y-Achse festzulegen. Beide Parameter

a_L und a_R werden durch einen Parameter a aus dem Intervall $I = [-1; 1]$ festgelegt. Wird ein a aus I gewählt, so ergibt sich der Bereich, aus dem die Werte für die Richtung stammen, gleichverteilt aus dem Intervall $I_a = [a_L; a_R]$ mit folgenden Werten für a_L und a_R:

$$a_L = \begin{cases} -1 & a \leq 0 \ , \\ 2a - 1 & 1 > a > 0 \ . \end{cases} \qquad a_R = \begin{cases} 1 & a \geq 0 \ , \\ 2a + 1 & 0 > a \geq -1 \ . \end{cases} \tag{5.28}$$

Durch die Wahl des Intervalls I_a wird erreicht, dass die Intervallbreite von I_a bei Wahl von a aus I an den Intervallgrenzen zu Null tendiert. Somit können Simulationen durchgeführt werden, in denen die Punkte ausschließlich in eine Richtung wandern. Erfolgt die Wahl des Parameters a aus dem Zentrum von I, so erreicht I_a die höchste Intervallbreite, womit eine völlig willkürliche Bewegung der Punkte erreicht werden kann.

Der Kreis um einen der Punkte entspricht dem Kommunikationsradius r_K, welcher in der Simulation für alle Teilnehmer den gleichen Wert besitzt. Diese Modellierung stellt eine Vereinfachung gegenüber der Realität dar. Bei Geräten variieren auf Grund ihrer Leistung i. A. die Werte für r_K, was aber im vorliegenden Fall vernachlässigt wird.

Die Festlegung, dass sich die Kommunikationsreichweite kreisförmig abgrenzt, ist ebenfalls eine von der Realität abweichende Annahme. Besonders innerhalb von Gebäuden existieren durch Streuung und Reflexionen gänzlich andere Strukturen.

Die Anforderung wurde eingeführt, um ein einfacheres Modell für den Aufbau der Simulationsumgebung zu erhalten. Mit der Forderung läuft die Kommunikation symmetrisch ab, d. h. kann ein Teilnehmer A Nachrichten an einen Teilnehmer B senden, so kann auch der Teilnehmer B Nachrichten an Teilnehmer A senden.

Ein Punkt kann mit allen sich innerhalb seines Kommunikationsradius befindlichen Punkten direkt kommunizieren. Für die Kommunikation über mehrere Punkte kann die Anzahl der Hops ebenfalls als Parameter festgelegt werden.

Es ist möglich Hindernisse in die Umgebung einzufügen. Mit jedem Hindernis sind Eigenschaften verbunden. So kann jedes eingezeichnete Hindernis anziehend, abstoßend oder neutral auf die mobilen Teilnehmer wirken. Die Wirkung beschränkt sich auf einen bestimmbaren Umkreis um das Hindernis. Kommt ein Punkt in den Einflussbereich eines Hindernisses, wird festgelegt, wie lange diese Wirkung andauern soll.

Mit diesem Mechanismus können Umgebungen realer nachgebildet werden als ohne die Einbettung von Hindernissen. So können beispielsweise Ticketschalter, Kioske oder Sitzgelegenheiten als anziehende und geschlossene Bereich oder Baustellen als abstoßende Objekte in die Simulationsumgebung eingezeichnet werden.

Im Auswahlfeld rechts unten in Abbildung 5.14 kann ausgewählt werden, ob ein lokales Gruppieren durchgeführt werden soll, und wenn dies geschehen soll, mit welcher Heuristik dies durchgeführt wird.

Neben diesem vorgestellten Bewegungsmodell existieren auch noch andere. Im *Random Waypoint Modell* wird ein Knoten zufällig in die Simulationsumgebung platziert und jeder Knoten wählt zufällig und gleichverteilt einen Zielpunkt und eine Geschwindigkeit. Ist der Zielpunkt erreicht, wiederholt sich der Vorgang, indem ein neuer Zielpunkt gewählt wird.

Bewegungsmodelle, wie sie u. a. Madsen *et al.* [MFP04] vorstellen, erlauben, dass sich der Geschwindigkeitsvektor $\vec{v}$ mit der Zeit ändert. Realisiert werden kann dieses Verhalten, indem beispielsweise in zeitlich definierten Abständen jeder Knoten seinen Geschwindig-

keitsvektor willkürlich ändert. Doch auch dieses Verfahren ist einem „Random Walk“ sehr ähnlich. Deshalb gibt es zusätzlich die Möglichkeit, mit jedem Punkt eine Funktion zu verknüpfen, die seine Bewegung beschreibt. Grundlage ist eine Funktion

$$\vec{v}(t) : \mathbb{R} \mapsto \mathbb{R} \times \mathbb{R} \text{ , d. h. } \vec{v}(t) = \begin{pmatrix} v_x(t) \\ v_y(t) \end{pmatrix}.$$

Da jedoch für jeden Knoten eine entsprechende Funktion definiert werden muss, ist das Verfahren sehr aufwändig.

Das vorliegende Bewegungsmodell der Brown'schen Bewegung wurde gewählt, da es für den Anwendungsbereich durchweg praxisgerecht ist. Wird die Bewegung der Personen in Fußgängerzonen, Bahnhofshallen oder Flughafenterminals in der Summe betrachtet, so bewegen sich Personen an solchen Ort sehr chaotisch, was der Brown'schen Bewegung entspricht.

Mit den zusätzlichen Eigenschaften, wie den Hindernissen, können solche Areale gut mit dem Bewegungsmodell nachgebildet werden.

5.6.2 Gruppenerstellung

Aus der Bewegung der Punkte, deren Simulation in Abbildung 5.14 zu sehen ist, sollen die Personen mit ähnlichen Profilen in Gruppen zusammengefasst werden. Abbildung 5.15 zeigt die Visualisierung gefundener Gruppen.

Abbildung 5.15: Simulationsumgebung für die Gruppierung

Im Gegensatz zu Abbildung 5.14 repräsentieren die Punkte in Abbildung 5.15 nicht die Personen selbst, sondern deren Profile, d. h. in diesem Anwendungsfall die Zielorte von Personen. Das kleine Quadrat links unten in Abbildung 5.15 stellt den Ausgangspunkt

der Gruppierung dar, d. h. sämtliche zu gruppierende Personen befinden sich in der Nähe dieses Punktes. Dieser Punkt entspricht der Simulationsumgebung aus Abbildung 5.14.

Die vier anderen Punkte symbolisieren die jeweiligen Zielorte von den vier Bewegungspunkten, die sich in Abbildung 5.14 innerhalb des Kommunikationsradius befinden. Drei der Profilpunkte sind mit einer Linie verbunden. Dies bedeutet, dass sie eine Gruppe bilden würden, da ihre Zielorte miteinander harmonieren. Der vierte Profilpunkt ist auf Grund seiner Abgelegenheit nicht mit in der Gruppe.

Die Route, die in obiger Abbildung zu sehen ist, ist rein schematisch durch Verbinden der Punkte entstanden und stellt keine real existierende Route dar.

5.7 Profilbasierte Gruppenführungen

Abschließend wird noch ein weiteres Szenario kurz betrachtet, um die restlichen Gruppierungsmöglichkeiten, die das MoPiDiG Framework bietet, nicht außer Acht zu lassen. Bei dem Szenario handelt es sich um profilbasierte Gruppenführungen. Nachdem das Szenario beschrieben ist, werden verschiedene Möglichkeiten der Modellierung gezeigt und wie diese in MoPiDiG integriert werden können.

5.7.1 Beschreibung des Szenarios

Das Szenario wurde bereits in Abschnitt 1.1.2.2 kurz erklärt. Ausgangspunkt ist eine kulturelle Einrichtung, in der Führungen abgehalten werden. Ziel ist die Qualität der Führung dadurch zu verbessern, dass die Führung auf die einzelnen Teilnehmer abgestimmt ist. Um zu verhindern, dass zu viele Führungsteilnehmer mit disjunkten Interessensgebieten an einer Führung teilnehmen, kann mit dem MoPiDiG Framework eine Interessengruppierung vorgenommen werden. Somit kann eine Führung erreicht werden, die nicht nur auf die Summe aller Teilnehmerinteressen abgestimmt ist, sondern es wird auch versucht, dass für jeden Teilnehmer möglichst wenige, für ihn unwesentliche, Themengebiete Teil der Führung sind.

5.7.2 Modellierung des Szenarios

Anhand der profilbasierten Gruppenführungen wird im weiteren Verlauf dieses Abschnittes erklärt, welche Möglichkeiten der Modellierung sich für dieses Szenario eigenen. Allen Modellen gemein ist jedoch das Benutzerprofil. In ihm sind die Sehenswürdigkeiten des Museums bzw. der kulturellen Einrichtung enthalten. Der Benutzer muss spezifizieren, welche Ausstellungsobjekte für ihn interessant sind und welche nicht.

5.7.2.1 Explizite ähnlichkeitsbasierte Modellierung

Als erste Möglichkeit wird die explizit ähnlichkeitsbasierte Modellierung vorgestellt. Der Benutzer sucht in diesem Fall nach Gruppen, indem er explizit die Themengebiete angibt, die die Teilnehmer der Gruppe in gleicher Weise interessant finden. Eine Gruppierungsbedingung bzw. eine Gruppierungsfunktion ist nicht notwendig.

5.7.2.2 Implizite ähnlichkeitsbasierte Modellierung

Bei der impliziten ähnlichkeitsbasierten Modellierung werden die ähnlichsten Benutzerprofile zu Gruppen zusammengeschlossen.

Um zu entscheiden, wie ähnlich sich Benutzerprofile sind, müssen Distanzfunktionen vorliegen. Am Beispiel der profilbasierten Führung sind sich Benutzerprofile ähnlich, wenn sich möglichst viele Interessen gleichen, d. h. eine Funktion, die die Anzahl an übereinstimmenden Einträgen ermittelt, könnte als Distanzfunktion dienen. Sind x und y zwei Profile, wird diese Eigenschaft von der Distanzfunktion

$$d(x,y) = \sum_{i=1}^{m} \delta(x_i, y_i) \qquad \text{mit } \delta(x_i, y_i) = \begin{cases} 0 & \text{falls } x_i = y_i, \\ 1 & \text{sonst.} \end{cases} \tag{5.29}$$

erfüllt.

Bei dieser Gruppierung ist zudem noch eine Gruppierungsfunktion notwendig, diese muss jedoch nicht vom Benutzer definiert werden, sondern ist bereits Teil des MoPiDiG Frameworks, so dass hierzu keine weitere Konfigurationsarbeit zu leisten ist.

5.7.2.3 Profitbasierte Modellierung

Neben den ähnlichkeitsbasierten Modellierungen kann die Anwendung der profilbasierten Führung jedoch ebenso wie das Taxi-Sharing Szenario als profitbasierte Gruppierung modelliert werden. Folglich sind eine Payoff-Funktion und eine Gruppierungsbedingung notwendig. Diese sind jedoch, wie im Folgenden gezeigt wird, sehr ähnlich zum Fall des Taxi-Sharing Szenarios.

Ausgangspunkt der profitbasierten Modellierung ist die Festsetzung eines Preises für die Führung. Es könnte beispielsweise ein Komplettpreis für jede Führung existieren, die den Besuch einer gewissen Anzahl an Ausstellungsobjekten umfasst. Um jedoch ein höheres Maß an Flexibilität bereitzustellen, besteht eine Führung aus der Erläuterung von m Ausstellungsobjekten, jede Erläuterung kostet einen Einheitspreis von p €. Der Gesamtpreis der Führung ist somit $m \cdot \ p$ €. Der Parameter m ist nur durch die Anzahl der Ausstellungsstücke nach oben beschränkt.

Der Preis P_1 für die Führung mit einer ersten Person, die die Ausstellungsstücke einer Menge M_1 begutachten will, ist gegeben durch

$$P_1 = |M_1| \cdot p,$$

wobei $|M_1|$ die Kardinalität der Menge M_1 bedeutet. Analog ergibt sich der Preis P_2 einer zweiten Person, die die Ausstellungsstücke einer Menge M_2 begutachten mit

$$P_2 = |M_2| \cdot p.$$

Der Gesamtpreis der beiden Führungen für die beiden Personen ergibt sich aus der Vereinigungsmenge der zu besichtigenden Ausstellungsstücke $M_1 \cup M_2$, womit sich der Preis P_{12} zu

$$P_{12} = |M_1 \cup M_2| \cdot p$$

ergibt.

Für n Personen mit den Interessensmengen $M_1, M_2, \ldots M_n$ ist somit der Gesamtpreis P_G durch

$$P_G = \left| \bigcup_{i=1}^{n} M_i \right| \cdot p$$

gegeben.

Payoff Funktion π: Die Payoff Funktion des Taxi-Sharing Szenarios kann auch für die profilbasierten Führungen verwendet werden. Aus diesem Grund sind die Gleichungen 5.1 und 5.2 hier noch einmal aufgelistet.

$$\pi(i) = P_S^i - P_{\mathcal{G}}^i \tag{5.30}$$

$$\pi(\mathcal{G}) = \sum_i \pi(i) \tag{5.31}$$

Die Nutzenfunktion ist damit die Differenz aus dem Preis der Einzelführung und Gruppenführung, jeweils pro Person. Der Nutzen der Gruppe ergibt sich aus der Summe der einzelnen Profite.

Gruppierungsbedingung: Generell muss für die Gruppierung

$$\pi(i) > 0$$

gelten. Da jedoch die Payoff Funktion π aus dem Taxi-Sharing-Szenario verwendet werden kann, ändert sich auch an der Gruppierungsbedingung nichts.

Preisverteilung: Die in Abschnitt 5.3.2 vorgestellten Preisverteilungstrategien können auch für dieses Szenario prinzipiell angewendet werden. Jedoch wird sich für profilbasierte Gruppenführung primär die Variante im Vordergrund stehen, dass der Gruppenpreis geteilt wird, da i. A. die Personen an der kompletten Führung teilnehmen.

Der Einzelpreis für jeden Gruppenteilnehmer beträgt somit i. A.

$$P_S = \frac{p}{n} \cdot \left| \bigcup_{i=1}^{n} M_i \right|.$$

Natürlich können auch die zwei anderen Preisvarianten eingesetzt werden. In diesem Fall ist jedoch notwendig, dass nur die bezahlte Information dem Führungsteilnehmer erhältlich ist. Realisierbar ist dies durchaus, soll jedoch im Weiteren nicht betrachtet werden.

Wie aus diesem Abschnitt hervorgeht, unterschiedet sich die profitbasierte Gruppierung in dieser Anwendung kaum vom Taxi-Sharing Szenario. Auf diese Weise können viele weitere Anwendungen modelliert werden. Voraussetzung ist, dass es einen Basispreis gibt, der anschließend auf mehrere Gruppenteilnehmer aufgeteilt wird. Die Gruppierungsbedingung muss in diesem Fall immer lauten, dass sich der Einzelpreis durch die Gruppe erniedrigen muss.

Eine weitere mögliche Anwendung ist noch die Ausnutzung von Gruppentarifen, d. h. dass sich eine Mindestanzahl an Personen finden muss, um beispielsweise einen günstigen Eintritt zu erlangen.

5.7.3 Domänenbeschreibung bei der Profilbasierten Gruppenführung

In diesem Abschnitt werden für die profilbasierte Gruppenführung die Profil- und Gruppenbeschreibung vorgestellt. Bei der Gruppenbeschreibung wird für jede Gruppierungsart ein Beispiel präsentiert.

5.7.3.1 Profilbeschreibung

In Abbildung 5.16 ist ein beispielhaftes Benutzerprofil für die profilbasierten Gruppenfüh-

```
<?xml version="1.0" standalone="yes" ?>
<USERPROFILE>
    <USERDATA>
        <USERID>123456789abc</USERID>
        <NAME>Max Mustermann</NAME>
    </USERDATA>
    <PROFILEENTRIES>
        <PROFILEENTRY>
            <ENTRYNAME>Expressionismus</ENTRYNAME>
            <ENTRYVALUE>1</ENTRYVALUE>
        </PROFILEENTRY>
        <PROFILEENTRY>
            <ENTRYNAME>Impressionismus</ENTRYNAME>
            <ENTRYVALUE>0</ENTRYVALUE>
        </PROFILEENTRY>

        ...

        <PROFILEENTRY>
            <ENTRYNAME>Surrealismus</ENTRYNAME>
            <ENTRYVALUE>1</ENTRYVALUE>
        </PROFILEENTRY>
    </PROFILEENTRIES>
</USERPROFILE>
```

Abbildung 5.16: Beispiel der Profilbeschreibung für die Profilbasierte Gruppenführung

rungsanwendung angegeben. Als Profileinträge wurden beispielhaft zufällige Stilrichtungen aus der Malerei ausgewählt. Eine Stilrichtung ist mit einer Eins markiert, wenn eine Person sich für diese Stilrichtung interessiert. Im anderen Fall ist ein Null eingetragen.

5.7.3.2 Gruppenbeschreibung

Nachdem die Profildefinition erklärt wurde, wird im Folgenden für jede Gruppierungsart die dazugehörige Gruppendefinition näher erläutert.

Explizit ähnlichkeitsbasierte Gruppierung: Abbildung 5.17 zeigt eine Gruppendefinition für den expliziten ähnlichkeitsbasierten Fall. Hier wird spezifiziert, dass alle Gruppenteilnehmer die Kategorien *Expressionismus* und *Surrealismus* als bedeutend in ihr Profil eingetragen haben, was durch die Eins im `ENTRYSTRINGVALUE` Feld angegeben

```
<?xml version="1.0" standalone="yes" ?>
<GROUPDESCRIPTION>
    <GROUPTYPE>explicit similar</GROUPTYPE>
    <ENTRIES>
        <ENTRY>
            <ENTRYNAME>Expressionismus</ENTRYNAME>
            <ENTRYSTRINGVALUE>1</ENTRYSTRINGVALUE>
        </ENTRY>
        <ENTRY>
            <ENTRYNAME>Surrealismus</ENTRYNAME>
            <ENTRYSTRINGVALUE>1</ENTRYSTRINGVALUE>
        </ENTRY>
    </ENTRIES>
    <GROUPSIZE>
        <LOWERLIMIT>1</LOWERLIMIT>
        <UPPERLIMIT>8</UPPERLIMIT>
    </GROUPSIZE>
    <VALIDITYPERIOD>
        <START>2005-07-23T10:00:00</START>
        <END>2005-07-23T12:00:00</END>
    <VALIDITYPERIOD>
</GROUPDESCRIPTION>
```

Abbildung 5.17: Beispiel für die explizit ähnlichkeitsbasierte Gruppenbeschreibung

wird. Alle Teilnehmer in der momentanen Kommunikationsreichweite, deren Benutzerprofil diese Charakteristik aufweist, gehören zur Gruppe dazu.

Als maximale Gruppengröße werden acht Personen zugelassen. Sollte es mehr potenzielle Gruppenmitglieder geben, entscheidet in der vorliegenden Implementierung der Zeitpunkt des Gruppenanschlusses. Somit werden die zeitlich gesehen ersten acht Personen, die den Kriterien der Gruppenbeschreibung genügen, zur Gruppe zusammengefügt.

Am Ende der Gruppenbeschreibung ist noch eine Gültigkeitsdauer spezifiziert. Diese hat jedoch die gleiche Bedeutung, wie bereits im Taxi-Sharing Szenario erwähnt wurde.

Implizit ähnlichkeitsbasierte Gruppierung: In Abbildung 5.18 ist die Gruppenbe-

```
<?xml version="1.0" standalone="yes" ?>
<GROUPDESCRIPTION>
    <GROUPTYPE>implicit similar</GROUPTYPE>
    ...
</GROUPDESCRIPTION>
```

Abbildung 5.18: Beispiel für eine Gruppenbeschreibung im implizit ähnlichkeitsbasierten Modell

schreibung für den implizit ähnlichkeitsbasierten Fall gegeben, was auch durch das Tag `GROUPTYPE` spezifiziert wird. In diesem Beispiel sollen die Teilnehmer zu Gruppen zusammengefügt werden, deren Profilbeschreibungen sich am ähnlichsten sind. Aus diesem Grund sind auch keine näheren Angaben über die Gruppierungseinträge notwendig. Es

werden von allen vorhandenen Teilnehmern sämtliche Einträge berücksichtigt und die Profile mit dem geringsten Abstand zu einer Gruppe zusammengeschlossen.

Da die Gruppengröße und die Gültigkeitsdauer keiner Änderung im Vergleich zum expliziten Fall bedürfen, wurden diese Einträge in Abbildung 5.18 nicht nochmals eingetragen.

Profitbasierte Gruppierung: Abschließend wird in Abbildung 5.19 noch die Grup-

```
<?xml version="1.0" standalone="yes" ?>
<GROUPDESCRIPTION>
    <GROUPTYPE>profit</GROUPTYPE>
    <ENTRIES>
        <ENTRYNAME>Expressionismus</ENTRYNAME>
        <ENTRYNAME>Surrealismus</ENTRYNAME>
        ...
    </ENTRIES>
    ...
</GROUPDESCRIPTION>
```

Abbildung 5.19: Beispiel für eine profitbasierte Gruppenbeschreibung

penbeschreibung für die nutzenbasierte Gruppierung gezeigt. Da sich jedoch die Struktur nicht von der des Taxi-Sharing-Szenarios unterscheidet, ist keine weitere Erläuterung notwendig.

5.8 Zusammenfassung

Gegenstand dieses Kapitels ist eine konkrete MoPiDiG Anwendung - das Taxi-Sharing-Szenario.

Im Taxi-Sharing-Szenario bilden sich Gruppen mit Personen, die ähnliche Fahrziele haben und sich deshalb ein Taxi teilen können. Ein Nutzen entsteht durch einen reduzierten Fahrpreis für die Gruppenmitglieder.

Für das Taxi-Sharing-Szenario kommen als Middleware mehrere Systeme in Betracht. Verwendung findet jedoch eine um ad hoc Funktionalitäten erweiterte LEAP-Plattform.

Danach werden das Taxi-Sharing-Szenario formal definiert und die Gruppierungsbedingungen angegeben. Ist eine Gruppe gefunden, muss jedem Teilnehmer ein Fahrpreis zugewiesen werden. Von den existierenden Möglichkeiten werden drei Fahrpreisbildungsvarianten intensiv untersucht. In einer ersten Variante wird der Gesamtfahrpreis gleichmäßig unter allen Gruppenteilnehmern aufgeteilt. Die zweite Variante teilt den Fahrpreis prozentual nach gefahrenen Streckenanteilen auf, während die dritte Variante für jeden Teilnehmer einen gleichen Einsparfaktor garantiert.

Im Taxi-Sharing-Szenario werden alle Algorithmen des MoPiDiG Frameworks implementiert. Neben den eigentlichen Gruppierungsverfahren wird auch die Segmentierung und Topologieerzeugung durchgeführt. Außerdem werden im Taxi-Sharing-Szenario Heuristiken verwendet. Um die Anzahl an Gruppierungsobjekten zu reduzieren werden Gebiete definiert, die - je nach Fahrpreisbildungsvariante - entweder einem Kreis oder einer El-

lipse entsprechen. Die Lokales-Minimum Heuristik wird verwendet, da der Suchraum sich hierdurch extrem verkleinert.

Die Domänenbeschreibung für das Taxi-Sharing-Szenario ist sehr einfach aufgebaut. Die Profildefinition besteht nur aus den Zielkoordinaten einer Person. Die Gruppendefinition gibt an, dass diese Koordinaten die Grundlage der Gruppierung darstellen.

Da das intensive Testen des Taxi-Sharing-Szenarios sehr ressourcenintensiv ist, ist eine Simulationsumgebung notwendig. Ziel der Simulationsumgebung ist, die Bewegung von potenziellen Teilnehmern möglichst realitätsnah nachzubilden, was durch die Einführung mehrerer Parameter geschieht.

Da im Taxi-Sharing-Szenario nicht sämtliche Möglichkeiten des MoPiDiG Frameworks benötigt werden, werden diese in einem weiteren Beispiel erläutert. Schwerpunkt ist die Beschreibung als ähnlichkeitsbasiertes Gruppierungsproblem. Als weiteres Szenario wird die profilbasierte Gruppenführung näher betrachtet.

Im nächsten Kapitel werden Ergebnisse präsentiert, die durch die Analyse des Taxi-Sharing-Szenarios erhalten wurden.

Kapitel 6

Evaluationsergebnisse

Nachdem in den letzten beiden Kapiteln sowohl der allgemeine Aufbau des MoPiDiG Frameworks als auch eine konkrete Anwendung im Detail besprochen wurden, werden in diesem Abschnitt Evaluationsergebnisse präsentiert. Ziel der Simulationen ist, Aussagen über die Anwendbarkeit von MoPiDiG zu treffen.

Zuerst wird die Kommunikationswahrscheinlichkeit für die Simulationsumgebung untersucht. Anschließend werden Simulationsdaten präsentiert, die durch intensive Untersuchung des Taxi-Sharing Szenarios erhalten wurden. Hier wird im Detail die Stabilität der virtuellen Topologie und deren Anwendungsgrenze betrachtet. Die Zeit, die eine lokale bzw. globale Gruppe stabil ist wird zuerst analytisch untersucht. Anschließend wird das analytische Ergebnis mit Simulationsergebnissen bestätigt.

Im Weiteren wird die Qualität der Heuristiken bewertet und abschließend wird die zeitliche Dauer eines Nachrichtenaustauschs mit mobilen Endgeräten überprüft um Aussagen zu erhalten, ob MoPiDiG hierfür geeignet ist.

6.1 Kommunikationswahrscheinlichkeit

Zuerst wird die Kommunikationswahrscheinlichkeit innerhalb der Simulationsumgebung analysiert. Für diese Untersuchung muss jedoch zwischen einem offenen und einem geschlossenen Simulationsbereich unterschieden werden.

Beide Varianten sind in Abbildung 6.1 einzusehen und werden nachfolgend erörtert. Ein Kreis innerhalb des Simulationsbereichs entspricht dem Kommunikationsbereich.

6.1.1 Offener Simulationsbereich

Ist der Simulationsbereich offen, so entspricht die Simulationsumgebung der Form einer Torusoberfläche, d. h. bewegt sich ein Knoten rechts aus dem Simulationsbereich heraus, betritt er auf der linken Seite sofort wieder den Simulationsbereich.

Die Wahrscheinlichkeit p_o, dass ein willkürlicher Knoten innerhalb der Simulationsumgebung mit Breite b und Länge l mit dem Knoten kommunizieren kann, beläuft sich folglich auf

$$p_o = \frac{r_K^2 \pi}{l \cdot b}, \tag{6.1}$$

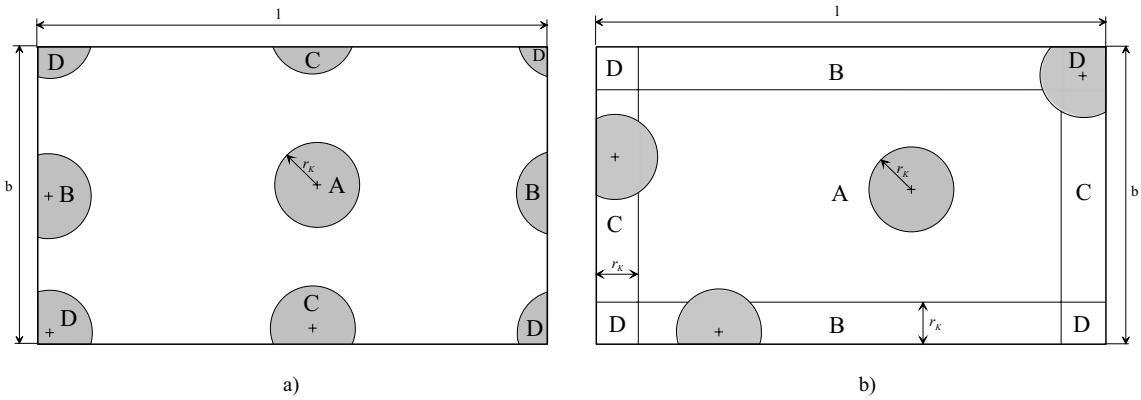


Abbildung 6.1: Offener (a) und geschlossener (b) Simulationsbereich

was das Verhältnis von Kreisfläche zu Rechtecksfläche darstellt.

Für den Erwartungswert E_o, der Anzahl an Kommunikationspartnern in einer offenen Simulationsumgebung gilt somit, wenn mit n die Anzahl der Knoten symbolisiert wird

$$E_o = (n-1) \cdot p_o. \tag{6.2}$$

Wird mit $\rho = \frac{n}{l \cdot b}$ die Knotendichte bezeichnet, wobei n die Anzahl der Elemente (= Knoten) in der Simulationsumgebung repräsentiert, so ergibt sich

$$E_o = (n-1) \cdot p_o = (n-1) \cdot \frac{r_K^2 \pi}{l \cdot b} = \rho r_K^2 \pi (1 - \frac{1}{n}) \approx \rho r_K^2 \pi. \tag{6.3}$$

6.1.2 Geschlossener Simulationsbereich

Ist der Simulationsbereich geschlossen, gestaltet sich die Berechnung der Kommunikationswahrscheinlichkeit wesentlich schwieriger. Wie in Abbildung 6.1b zu sehen ist, existieren in diesem Fall Bereiche, in denen keine kompletten Kreise als Kommunikationsbereich vorliegen.

Bereich A ist der Bereich, in welchem der Kommunikationsbereich vollständig durch Kreise mit Radius r_K angegeben werden kann.

In den Bereichen B und C beläuft sich die Flächengröße zwischen einem Halb- und einem Vollkreis. Ziel ist es, einen durchschnittlichen, gegenüber r_K reduzierten, Radius r_{BC} anzugeben, mit dem in diesen Bereichen als Ersatz für r_K gerechnet werden kann. Die Fläche resultiert aus der Fläche eines Vollkreises abzüglich der Fläche eines Kreisabschnitts.

Mit den in Abbildung 6.2a eingeführten Bezeichnungen für φ, y und g gilt [BSM00] für die Kreisabschnittsfläche A_{KA}

$$A_{KA} = r_K^2 \varphi - \frac{1}{2} g y = r_K^2 \arccos \frac{y}{r} - y\sqrt{r^2 - y^2}. \tag{6.4}$$

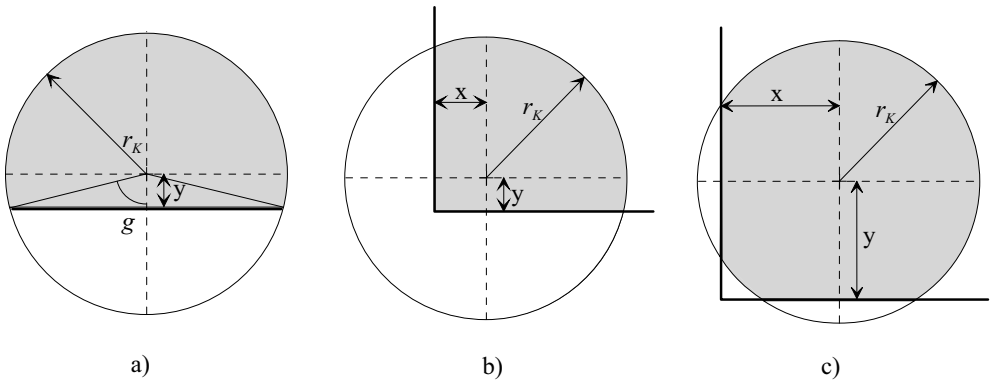


Abbildung 6.2: Geometrie des Kreisausschnitts

Mit Gleichung 6.4 ergibt sich für die Bereichsfläche in Abhängigkeit von der Randentfernung y

$$A_{CD}(y) = r_K^2\pi - r_K^2 \arccos\frac{y}{r_K} + y\sqrt{r_K^2 - y^2}. \tag{6.5}$$

Um den Mittelwert für den Radius in den Bereichen B und C zu erhalten, muss Gleichung 6.5 integriert werden, womit sich

$$\int_0^r A_{CD}(y)\, dy = r_K^3(\pi - \frac{2}{3}) \tag{6.6}$$

ergibt.

Mit diesem Ergebnis kann der Flächenreduktionsfaktor angegeben werden:

$$\frac{r_K^3(\pi - \frac{2}{3})}{r_K \cdot r_K^2\pi} = 1 - \frac{2}{3\pi} = k_{BC} \approx 0.7878. \tag{6.7}$$

Somit kann in den Bereichen B und C mit einer reduzierten Kreisfläche $k_{BC} \cdot r_K^2\pi$, mit $k_{BC} \approx 0.7878$ gerechnet werden.

In den vier Eckbereichen (Bereich D) muss ebenso wie bei den Bereichen B und C verfahren werden, wobei die Fläche in diesem Fall zwischen einem Viertel- und einem Vollkreis liegt. Ziel ist es, einen Flächenreduktionsfaktor k_D anzugeben.

Hinsichtlich der Flächenberechnung müssen zwei Fälle unterschieden werden. Im ersten Fall besteht die Bereichsfläche aus einem Viertelkreis, aus zwei Viertelkreisen abzüglich eines Kreisabschnitts und einem Rechteck $(x \cdot y)$. In Abbildung 6.2b ist dies ersichtlich.

Im zweiten Fall ergibt sich die Bereichsfläche aus einem Vollkreis abzüglich zweier

Kreisabschnitte, wie in Abbildung 6.2c zu sehen ist. Somit ergeben sich folgende Flächen

$$\begin{aligned} A_D^1(x,y) &= \frac{3}{4}r_K^2\pi - \frac{1}{2}r_K^2(\arccos\frac{y}{r} + \arccos\frac{x}{r}) + \\ &\quad \frac{1}{2}y\sqrt{r^2-y^2} + \frac{1}{2}x\sqrt{r^2-x^2}, + xy \qquad (6.8) \\ A_D^2(x,y) &= r_K^2 - r_K^2(\arccos\frac{y}{r} + \arccos\frac{x}{r}) + \\ &\quad y\sqrt{r^2-y^2} + x\sqrt{r^2-x^2}. \qquad (6.9) \end{aligned}$$

Um die durchschnittliche Kreisfläche zu ermitteln, muss über die Flächen $A_D^1(x,y)$ und $A_D^2(x,y)$ integriert werden. Hierbei ist zu beachten, dass $A_D^1(x,y)$ nur berücksichtigt wird, wenn sich die Simulationsumgebungsbegrenzung, d. h. die Ecke in Abbildung 6.2c, innerhalb des Kreises befindet. Für die restliche Fläche wird $A_D^2(x,y)$ verwendet. Somit ergibt sich

$$\int_0^r \int_0^{\sqrt{r_K^2-y^2}} A_1(x,y)\,dx\,dy + \int_0^r \int_{\sqrt{r_K^2-y^2}}^r A_2(x,y)\,dx\,dy = r^4(\pi - \frac{29}{24}). \qquad (6.10)$$

Für den Flächenreduktionsfaktor r_D ergibt sich somit

$$\frac{r^4(\pi - \frac{29}{24})}{r_K^2 \cdot r_K^2\pi} = 1 - \frac{29}{24\pi} = k_D \approx 0,6154. \qquad (6.11)$$

Folglich ist im Bereich D mit einer durchschnittlichen Fläche von $k_D \cdot r_K^2\pi$ zu rechnen.

Zur Berechnung der Wahrscheinlichkeiten sind sämtliche oben errechnete Bereiche in entsprechender Gewichtung mit der Gesamtfläche in Relation zu setzen. Somit ergibt sich

$$p_g = \frac{r_K^2\pi}{lb}\left(\frac{(l-2r_K)(b-2r_K)}{lb} + \frac{2r_K(l+b-4r_K)}{lb}k_{BC} + \frac{4r_K^2}{lb}k_D\right). \qquad (6.12)$$

Der Erwartungswert E_g der Anzahl der Kommunikationspartner in einer geschlossenen Simulationsumgebung beträgt - ähnlich zu Gleichung 6.2

$$E_g = (n-1)\cdot p_g. \qquad (6.13)$$

Die Simulation wurde mit den Daten $l = 600m$, $b = 350m$ und $r_K = 50m$ durchgeführt, so dass sich für die Wahrscheinlichkeiten folgende Werte ergeben

$$p_o = 0,0373999, \qquad (6.14)$$
$$p_g = 0,0338809. \qquad (6.15)$$

Die beiden Werte unterscheiden sich zwar absolut nur sehr geringfügig voneinander, prozentual gesehen beträgt der Unterschied jedoch 10%. Dieser Wert gilt somit auch für den Erwartungswert.

Wird für die Kantenwahrscheinlichkeiten p_o und p_g aus den die Zusammenhangswahrscheinlichkeit P_n gemäß Gleichung 3.6 rekursiv ermittelt, ergeben sich die in Tabelle 6.1 angegebenen Werte. Zu erkennen ist, dass anfänglich die Wahrscheinlichkeit des Zusammenhangs nicht gegeben ist, was hinsichtlich der geringen Werte für p_o und p_g auch nicht anders zu erwarten ist. Jedoch bereits ab einer Knotenanzahl von 200 kann davon ausgegangen werden, dass ein im Netz ein Zusammenhang vorliegt.

	p_o	p_g
5	$2{,}09 \cdot 10^{-4}$	$1{,}42 \cdot 10^{-4}$
25	$1{,}87 \cdot 10^{-5}$	$3{,}57 \cdot 10^{-7}$
100	0,105	0,021
150	0,621	0,419
200	0,910	0,814
250	0,981	0,951
250	-	0,991

Tabelle 6.1: Entwicklung der Zusammenhangswahrscheinlichkeit.

6.2 Simulationsergebnisse

In diesem Abschnitt werden die Simulationsergebnisse präsentiert, die durch Untersuchung des Taxi-Sharing Szenarios erhalten wurden.

Die Simulationsergebnisse wurden mit der Simulationsumgebung erzeugt, die in Abschnitt 5.6 im Detail erläutert wird. Um die Diagramme in den folgenden Abschnitten zu erhalten, wurde an die Simulationsumgebung eine Messdatenerfassung angebunden. Diese Messdatenerfassung protokolliert alle 200 Millisekunden den Zustand der Simulationsumgebung, d. h. alle 200 Millisekunden werden in eine Datei die für die jeweilige Auswertung wesentlichen Daten (z. B. Anzahl an Kommunikationspartner für jeden Punkt) geschrieben. Ein Simulationsdurchgang dauerte 15 Minuten, so dass nach dieser Zeit 4500 Datensätze zur Verfügung standen. Um repräsentative Daten zu erhalten, wurden mehrere Simulationsdurchgänge durchgeführt.

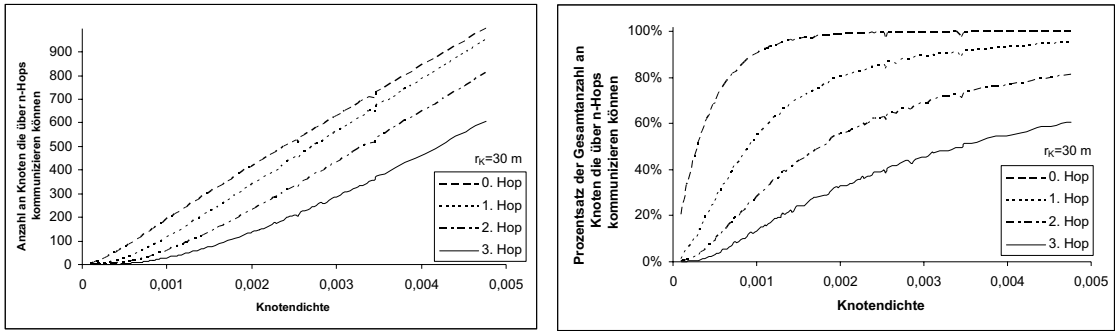
Ein zusätzliche Auswertungskomponente erzeugte aus den Datensätzen Tabellen, die in die nachstehenden Diagramme umgewandelt werden können.

6.2.1 Anzahl an Kommunikationspartnern

Für die Gruppenbildung wird ein ad hoc Netzwerk in Segmente aufgeteilt. Parameter dieser Segmentierung ist ein Wert k, der die Anzahl an Hops ausgehend von einem Initiator entspricht. Damit zwei Knoten jedoch über n-Hops miteinander kommunizieren können, müssen die Gegebenheiten, d. h. eine ausreichende Anzahl an potenziellen Router-Knoten, vorhanden sein. Hierüber geben die Abbildung 6.3 für einen Kommunikationsradius von 30 Metern und Abbildung 6.4 für einen Kommunikationsradius von 50 Metern Auskunft. Hierbei zeigen Abbildung 6.3a und 6.4a die absolute Anzahl an Knoten, die über eine gewisse Hop-Anzahl kommunizieren können und Abbildung 6.3b und 6.4b den prozentualen Anteil, gemessen an der absoluten Knotenanzahl, jeweils in Abhängigkeit der Knotendichte. Die Knotendichte ergibt sich in der Simulationsumgebung aus dem Verhältnis der Knotenanzahl, die einen Parameter der Simulationsumgebung darstellt und der Flächengröße, die für alle Untersuchungen eine konstante Größe aufweist.

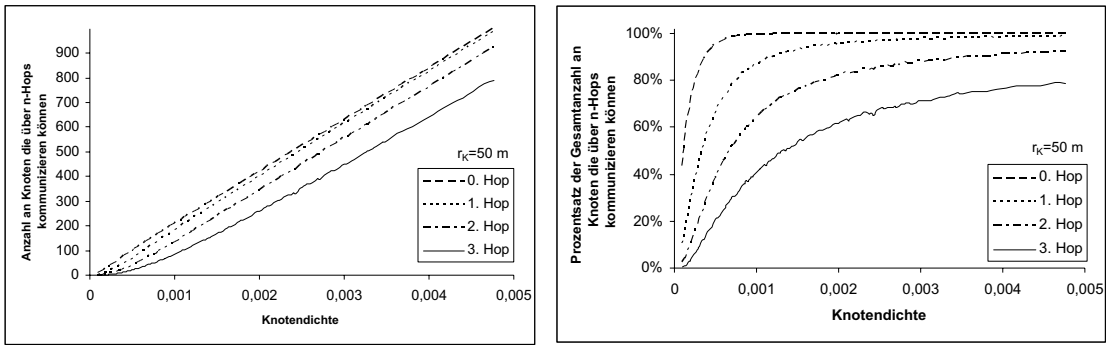
In allen obigen Abbildung entspricht eine 0-Hop Kommunikation einer direkten Kommunikation, d. h. Knoten befinden sich innerhalb des Kommunikationsradius r_K und es ist kein Knoten, der Routerfunktionalität leistet, für die Kommunikation notwendig.

Wie zu erwarten, nimmt mit zunehmender Knotendichte die Wahrscheinlichkeit über mehrere Hops kommunizieren zu können zu. Ferner nimmt mit zunehmendem Radius die



a) b)

Abbildung 6.3: Absolute Anzahl der Knoten, die über n-Hops kommunizieren können in Abhängigkeit von der Knotendichte (a). Bild b) zeigt den Prozentsatz der Knoten, die über n-Hops kommunizieren können ($r_K = 30$m).



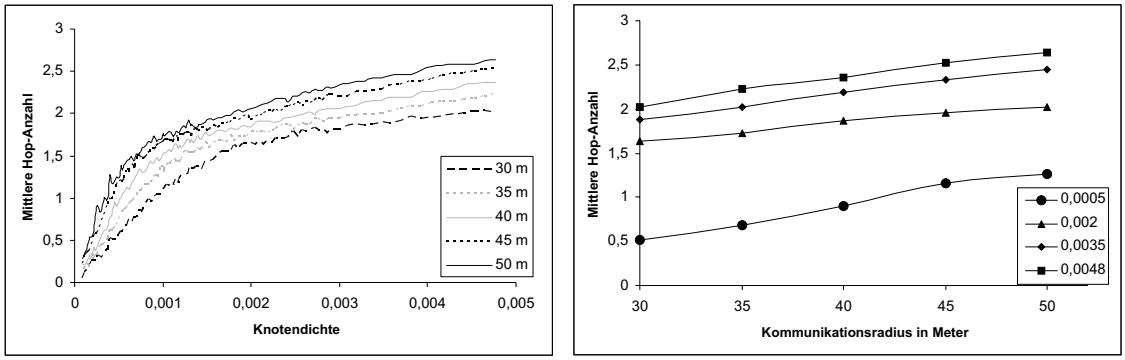
a) b)

Abbildung 6.4: Absolute Anzahl der Knoten, die über n-Hops kommunizieren können in Abhängigkeit von der Knotendichte (a). Bild b) zeigt den Prozentsatz der Knoten, die über n-Hops kommunizieren können ($r_K = 50$m).

Wahrscheinlichkeit für eine n-Hop Kommunikation zu.

Obige Diagramme zeigen, wie wahrscheinlich generell eine n-Hop Kommunikation für einen Knoten ist. Abbildung 6.5a stellt hingegen die mittlere Hop-Anzahl bei einer gegebenen Knotendichte für Kommunikationsradien r_K von 30m bis 50m dar. Hieraus lässt sich erkennen, dass eine mittlere Hop-Anzahl von zwei relativ schnell erreicht wird, der Anstieg dann jedoch nachlässt. Somit kann eine 2-Hop Kommunikation durchaus für die Anwendung noch in Betracht gezogen werden, eine 3-Hop Kommunikation ist nur bei sehr hohen Knotendichten überhaupt realisierbar.

Abbildung 6.5b zeigt die mittlere Hop-Anzahl in Abhängigkeit vom Kommunikationsradius r_K für ausgewählte Knotendichten und lässt einen quasi-linearen Anstieg der



a) b)

Abbildung 6.5: Entwicklung der Hop-Anzahl in Abhängigkeit von der Knotendichte bei unterschiedlichen Kommunikationsradien r_K (a) und in Abhängigkeit vom Kommunikationsradius bei verschiedenen Knotendichten (b)

mittleren Hop-Anzahl mit zunehmendem Kommunikationsradius erkennen.

Gemäß der vorherigen Untersuchungen ist eine 2-Hop Kommunikation topologisch möglich. Für die Gruppenbildung ist jedoch auch notwendig zu wissen, wie viele Knoten mit einer n-Hop Kommunikation am Gruppierungsprozess teilnehmen können. Über die Abhängigkeit der Segmentgröße von der vorherrschenden Knotendichte informiert Abbildung 6.6. Dort ist die Entwicklung sowohl für eine 1-Hop, als auch für eine 2-Hop

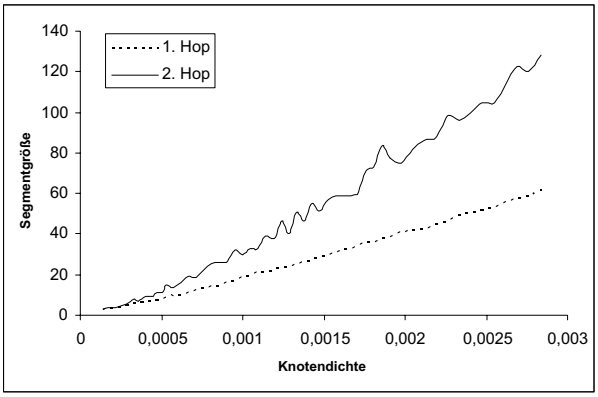


Abbildung 6.6: Anstieg der Segmentgröße in Abhängigkeit von der Knotendichte (r_K=50m)

Kommunikation dargestellt. Aus dem Diagramm in Abbildung 6.6 ist ersichtlich, dass bereits bei der 1-Hop Kommunikation bei einer Knotendichte von $0,0025$ eine Segmentgröße von mehr als 40 Teilnehmern erreicht ist. Wird eine 2-Hop Kommunikation durchgeführt,

so verdoppelt sich die Segmentgröße im Vergleich zur 1-Hop Kommunikation. Aus diesen Daten kann somit gefolgert werden, dass eine 1-Hop-Kommunikation i. A. ausreichen wird, um eine genügend große Basis an Teilnehmern für eine Gruppenbildung zur Verfügung zu haben.

Die Anzahl an Kommunikationspartnern für eine bestimmte Anzahl an Hops kann auch analytisch abgeschätzt werden.

Im Falle vorliegender 1-Hop Kommunikation beträgt der maximale Kommunikationsradius den doppelten Wert, wie in Abbildung 6.7b zu sehen ist. Da jedoch dieser maximale

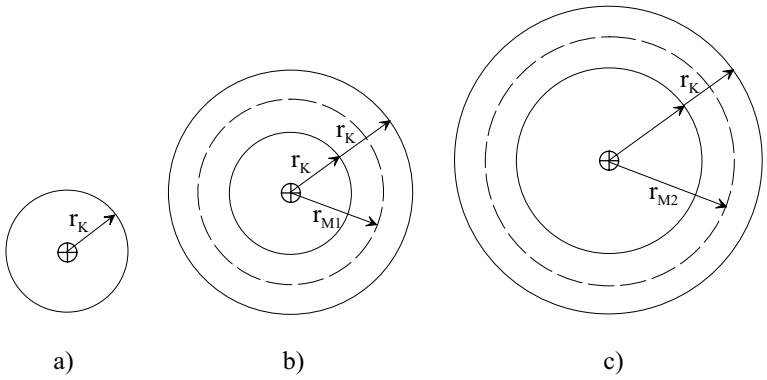


Abbildung 6.7: a) zeigt den direkten Kommunikationsbereich. In b) ist der reduzierte 1-Hop und in c) der reduzierte 2-Hop Kommunikationsbereich mit den Radien r_{M1} bzw. r_{M2} gestrichelt dargestellt.

Kommunikationsradius nur bei hinreichend vielen Knoten innerhalb des 0-Hop Kommunikationsbereiches zu Stande kommt, soll im Folgenden ein mittlerer Radius r_{M1} für die 1-Hop Kommunikation ermittelt werden.

Ausgangspunkt ist hierfür ein Kreis mit dem Radius r_K, der auf $2r_K$ vergrößert wird (siehe Abbildungen 6.7a und 6.7b), um auf diese Weise den mittleren Radius zu bestimmen.

Wird ein Kreis mit der Fläche A_{K1} und dem Radius r_K, um ein Δx vergrößert, ergibt sich

$$A_{K1} = (r_K + \Delta x)^2 \pi \;\; (r_K \geq \Delta x \geq 0),$$

Für den mittleren Flächeninhalt $\bar{A}_{K1}$ ergibt sich aus

$$\bar{A}_{K1} = \frac{1}{r} \int_0^r A_{K1} \; dx = \frac{7}{3} r_K^2 \pi.$$

Somit kann der mittlere Radius r_{M1} bestimmt werden:

$$\begin{aligned} r_{M1}^2 \pi &= \frac{7}{3} r_K^2 \pi \Rightarrow \\ r_{M1} &= \frac{1}{3}\sqrt{21} \cdot r_K. \end{aligned} \tag{6.18}$$

Der Wert von r_{M1} entspricht somit ungefähr $1,528 \cdot r_K$. Um die Anzahl an Kommunikationsteilnehmern abzuschätzen, muss nur die Anzahl an Punkten in diesem Kommunika-

tionsbereich bestimmt werden, was durch

$$E_{r_{M1}} = r_{M1}^2 \pi \rho \tag{6.19}$$

geschehen kann.

Für eine 2-Hop Kommunikation gilt entsprechendes, nur dass hier ein Kreis mit dem Radius r_{M1}, um r_K vergrößert wird, wie auch in Abbildungen 6.7c zu sehen. Folglich ist

$$A_{K2} = (r_{M1} + x)^2 \pi = (\frac{1}{3}\sqrt{21} r_K + x)^2 \pi \;\; (r_K \geq \Delta x \geq 0),$$

und es ergibt sich der Mittelwert

$$\bar{A}_{K2} = \frac{1}{r} \int_0^r A_{K2} \; dx = \frac{1}{3} r_K^2 \pi (8 + \sqrt{21}).$$

Als mittlerer Radius r_{M2} ergibt sich

$$\begin{aligned} r_{M2}^2 \pi &= \frac{1}{3} r_K^2 \pi (8 + \sqrt{21}) \Rightarrow \\ r_{M2} &= \frac{1}{3}\sqrt{3}\sqrt{8 + \sqrt{21}} \cdot r_K. \end{aligned} \tag{6.22}$$

Für den Erwartungswert für die Anzahl der Kommunikationspartner gilt

$$E_{r_{M2}} = r_{M2}^2 \pi \rho. \tag{6.23}$$

Für niedrige Knotendichten stellen die Erwartungswerte in den Gleichungen 6.19 und 6.23 obere Grenzen dar, die in der Regel nicht erfüllt wird. Die Berechnung liefert die Anzahl an Punkten in dem Kreisgebiet. Für eine n-Hop Kommunikation ist jedoch Voraussetzung, dass sich $(n-1)$ Knoten in jeweiliger Kommunikationsreichweite befinden. Bei niedrigen Knotendichten kann hiervon nicht ausgegangen werden, so dass in diesem Fall der analytische Wert höher liegt als der tatsächliche.

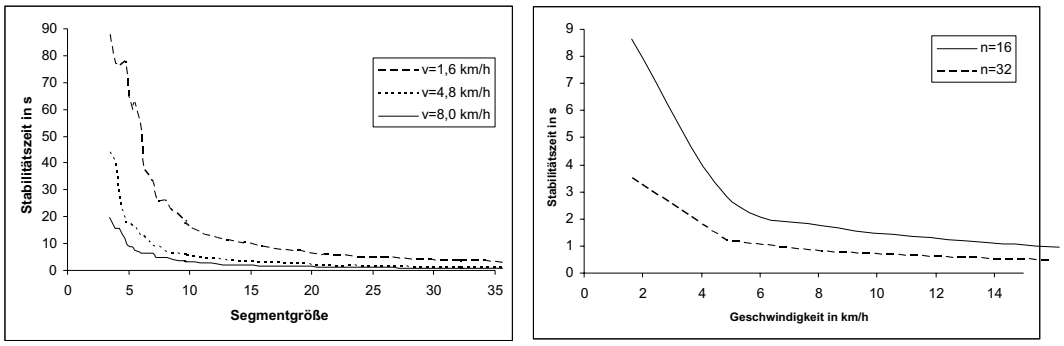
6.2.2 Virtuelle Topologie

In diesem Abschnitt werden Simulationsergebnisse präsentiert, die Aussagen über die Verwendbarkeit der virtuellen Topologie möglich machen. Zuerst wird die Baumtopologie untersucht, die anschließend mit der Ringtopologie verglichen wird.

6.2.2.1 Baum

Bei der virtuellen Topologie ist primär die Stabilitätszeit t_s^T von Interesse. Unter der Stabilitätszeit t_s^T sei die Zeitspanne verstanden, die verstreichen kann, ohne dass Teilnehmer der Topologie diese verlassen oder neue Teilnehmer hinzukommen und gleichzeitig keine internen Änderungen auftreten.

Abbildung 6.8a zeigt die Stabilitätszeit in Abhängigkeit der Segmentgröße für verschiedene Geschwindigkeiten v. Die angegebenen Werte für die Geschwindigkeit v sind mittlere Geschwindigkeiten, d. h. ist eine Geschwindigkeit von v_i angegeben, so werden für die Simulation Geschwindigkeiten aus dem Intervall $[0, 2 \cdot v_i]$ gleichverteilt gewählt.



a) b)

Abbildung 6.8: Bild a) zeigt die Stabilität der Baumstruktur in Abhängigkeit von der Segmentgröße für verschiedene Geschwindigkeiten. In Bild b) ist die Stabilität in Abhängigkeit von der Geschwindigkeit dargestellt.

Abbildung 6.8a zeigt die rasche Abnahme der Stabilitätszeit t_s^T für eine Baumtopologie bei gegebener Segmentgröße. Bei einer mittleren Geschwindigkeit von $1,6$ km/h ist eine Baumtopologie bei einer Segmentgröße von 15 Knoten für zehn Sekunden stabil. Bei einer Geschwindigkeit von $4,8$ km/h wird dieser Wert bereits bei sieben Knoten und bei $8,0$ km/h bei fünf Knoten erreicht. Ab einer Segmentgröße von 25 Knoten bewegt sich für alle untersuchten Geschwindigkeiten die Stabilitätszeit im Sekundenbereich.

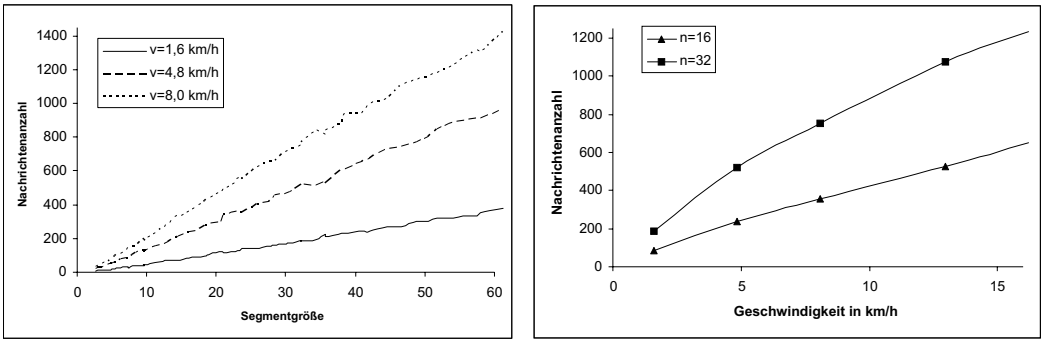
Abbildung 6.8b veranschaulicht die Stabilitätszeit t_s^T in Abhängigkeit von der Geschwindigkeit v für zwei verschiedene Segmentgrößen. Die Abnahme von t_s^T erfolgt ebenfalls sehr schnell, wobei sich die Abnahme ab einer Geschwindigkeit von ~ 2 km/h reduziert.

Die ausschließliche Betrachtung der Stabilitätszeit t_s^T sagt noch nichts über die Einsetzbarkeit der Topologie aus. Die Stabilitätszeit gibt die Zeitspanne an, in der sich die Knotenanzahl der Topologie nicht ändert. Erfolgt jedoch eine Änderung, sind zusätzlich Nachrichten notwendig um die Topologie aufrecht zu erhalten. Im Folgenden wird die Anzahl der Änderungsnachrichten untersucht. Grundlage der Simulation waren jeweils Simulationseinheiten mit einer Dauer von 300 Sekunden, so dass sich die Anzahl an Nachrichten explizit auf diesen Zeitraum bezieht.

Abbildung 6.9a stellt deswegen die Anzahl an Nachrichten, die notwendig sind die Baumtopologie aufrechtzuerhalten, in Abhängigkeit von der Segmentgröße für verschiedene mittlere Geschwindigkeiten v dar. Der Anstieg der Nachrichten steigt für niedrige Geschwindigkeiten linear an. Für höhere Geschwindigkeiten lässt sich mit zunehmender Segmentgröße ein geringerer Anstieg der Nachrichten ablesen.

In Abbildung 6.9b ist die Anzahl an Topologieaufrechterhaltungsnachrichten in Abhängigkeit der Geschwindigkeit für ausgewählte Segmentgrößen wiedergegeben.

Hinsichtlich der Topologieänderungen kann noch hinzugefügt werden, dass im Durchschnitt mehr Knoten das Segment/die Topologie verlassen als neue hinzukommen. Die Ursache ist darin zu suchen, dass wenn ein Knoten aus dem Segment verschwindet, noch



a) b)

Abbildung 6.9: Bild a) zeigt die Anzahl an Topologieaufrechterhaltungsnachrichten für die Baumstruktur in Abhängigkeit von der Segmentgröße. Bild b) bildet die Abhängigkeit von der Geschwindigkeit ab.

weitere Knoten, die mit diesem verbunden sind, das Segment verlassen. Will ein Knoten sich dem Segment anschließen, ist dies natürlich möglich, jedoch nur für einzelne Knoten. Wollen sich mehrere Knoten dem Segment anschließen, ist dies i. A. nicht möglich, da in diesem Fall der für das Segment erlaubte maximale Durchmesser überschritten wird.

Mit der Simulationsumgebung ist es möglich die Bewegung der Knoten derart zu parametrisieren, dass sich unterschiedliche Bewegungsmuster ergeben. Bei den bisherigen vorgestellten Untersuchungen bewegten sich die Knoten völlig willkürlich, d. h. gemäß einer Brown'schen Bewegung (siehe Abschnitt 5.6.1). Im Folgenden bewegen sich die Knoten mit einer Vorzugsrichtung, d. h. die Bewegung erfolgt entlang einer Dimension, wie es in z. B. in Fußgängerzonen vorherrschend ist.

Abbildung 6.10 zeigt die Stabilitätszeit t_s^T in Abhängigkeit von der Segmentgröße sowohl für eine willkürliche Bewegung als für eine Bewegung mit Vorzugsrichtung. Zu erkennen ist, dass - obwohl sich das Bewegungsmuster unterscheidet - kein nennenswerter Unterschied hinsichtlich der Stabilitätszeit t_s besteht.

6.2.2.2 Ring

Nachdem im letzten Abschnitt die Baumtopologie untersucht wurde, erfolgt in diesem Abschnitt die Beschreibung der Ergebnisse für die Ringtopologie.

Wie bei der Baumtopologie wird zuerst die Stabilitätszeit betrachtet. Abbildung 6.11a zeigt die Stabilitätszeit in Abhängigkeit der Segmentgröße. Wie bereits bei der Baumtopologie erfolgt die Abnahme sehr rapide. Ein Segment mit 20 Knoten ist bei einer Geschwindigkeit von $1,6$ km/h für zehn Sekunden stabil. Bei einer Geschwindigkeit von $4,8$ km/h wird diese Zeit noch bei einer Segmentgröße von acht Knoten und bei einer Geschwindigkeit von $8,0$ km/h noch bei einer Größe von fünf Knoten erreicht.

In Abbildung 6.11b ist die Stabilitätszeit in Abhängigkeit der Geschwindigkeit aufgezeigt. Der Kurvenverlauf ist ähnlich dem in Abbildung 6.8b. Jedoch beträgt bei einer

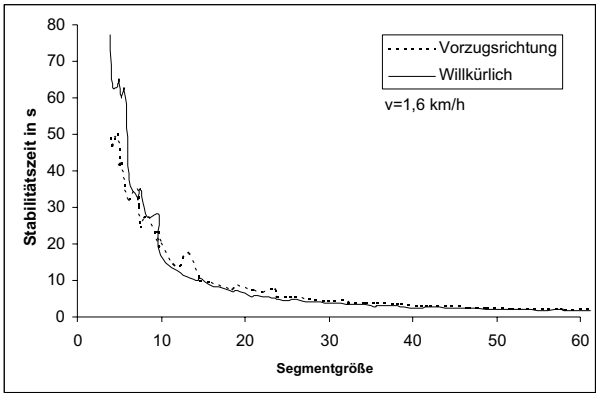
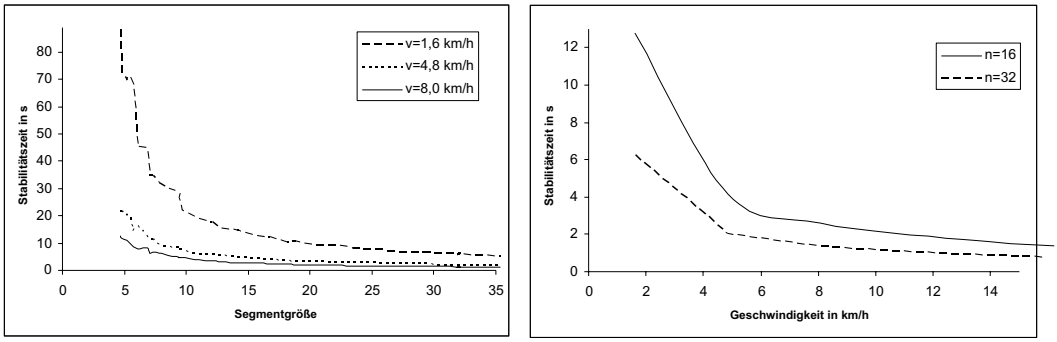


Abbildung 6.10: Stabilität der Topologie bei Vorzugsrichtung der Knoten und willkürlicher Bewegung

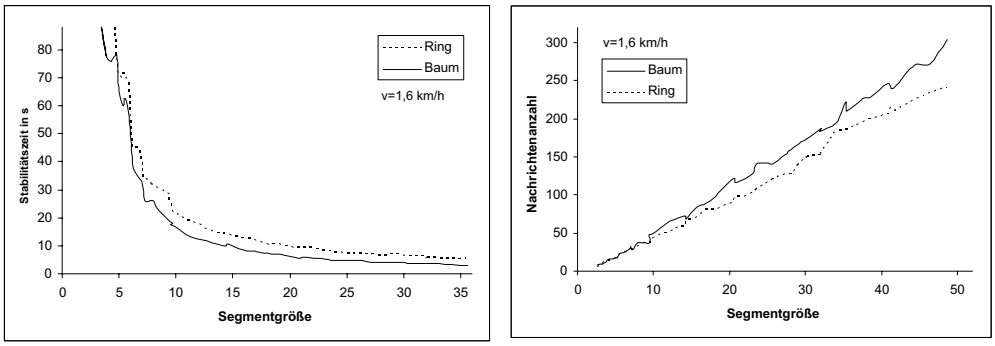


a) b)

Abbildung 6.11: Bild a) zeigt die Stabilität der Ringstruktur in Abhängigkeit von der Segmentgröße für verschiedene Geschwindigkeiten. In Bild b) ist die Stabilität in Abhängigkeit von der Geschwindigkeit dargestellt.

Ringtopologie bei einer Geschwindigkeit von fast 5 km/h die Stabilitätszeit $\sim$ neun Sekunden, während bei einer Baumtopologie diese nur $\sim$ sechs Sekunden beträgt.

Um die beiden virtuellen Topologien besser miteinander vergleichen zu können, sind in Abbildung 6.12 Ergebnisse beider gegenübergestellt. Abbildung 6.12a vergleicht die Stabilitätszeit der Baum- mit der der Ringtopologie. Anfänglich weisen beide Topologien die gleiche Stabilitätszeit auf. Ab einer Segmentgröße von $\sim$ zehn Knoten entwickelt die Ringtopologie eine etwas höhere Stabilitätszeit. Dies spiegelt sich auch in der Anzahl der Topologieaufrechterhaltungsnachrichten wider, welche in Abbildung 6.12b für beide Topologien dargestellt sind. Auch in diesem Fall werden bei der Ringtopologie ab einer Segmentgröße von $\sim$ zehn Knoten weniger Nachrichten benötigt um die Topologie auf-



a) b)

Abbildung 6.12: Bild a) stellt die Stabilitätszeit der Baumtopologie der der Ringtopologie gegenüber. Bild b) vergleicht die Anzahl an Nachrichten zur Aufrechterhaltung der Topologie

rechtzuerhalten als bei der Baumtopologie.

6.2.3 Gruppenbildung

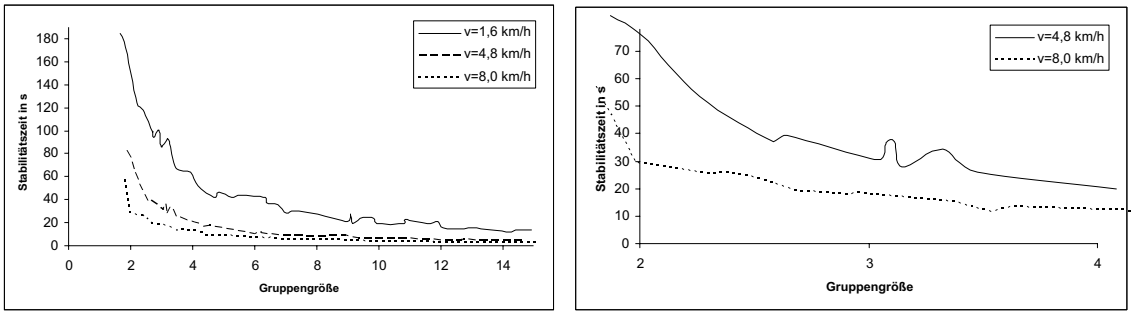
Nachfolgend werden Simulationsergebnisse der Gruppenbildung präsentiert. Kern der Untersuchungen ist die Stabilitätszeit und die Größe von lokalen und globalen Gruppen.

6.2.3.1 Lokale Gruppenbildung

Wie bereits bei der Untersuchung der virtuellen Topologie ist auch bei der Gruppenbildung die Stabilitätszeit ein wesentliches Merkmal. Unter der Stabilitätszeit t_s^G bei der Gruppenbildung sei die Zeitspanne verstanden, die verstreicht, ohne dass Teilnehmer der Gruppe diese verlassen oder neue Teilnehmer hinzukommen. Die Stabilitätszeit ist somit deswegen von Bedeutung, weil innerhalb dieser Zeitspanne keine Aktualisierungsmechanismen notwendig sind.

Die Abbildungen 6.13a und 6.13b zeigen die Stabilitätszeit t_s^G einer lokalen Gruppe in Abhängigkeit der Gruppengröße für verschiedene mittlere Geschwindigkeiten. Während Abbildungen 6.13a den kompletten Kurvenverlauf bis zu sehr großen Gruppen wiedergibt, konzentriert sich Abbildung 6.13b auf Gruppengrößen bis vier Teilnehmer, da für das Taxi-Sharing Szenario größere Gruppen i. A. nicht in Betracht kommen. Dort ist zu sehen, dass selbst für eine durchschnittliche Geschwindigkeit von 8 km/h eine Gruppe mit vier Personen für annähernd zehn Sekunden stabil ist. Die Abhängigkeit für noch größere mittlere Geschwindigkeit als 8 km/h wurde nicht untersucht, da dies bereits eine obere Grenze für Geschwindigkeit von Fußgängern darstellt.

Die Abhängigkeit der Stabilitätszeit $t_n^{s_\ell}$ lokaler Gruppen von der Gruppengröße n wird im Folgenden analytisch betrachtet. Zuerst wird die Zeit $t_2^{s_\ell}$ betrachtet, die eine lokale Gruppe mit einem Zusatzpunkt - also einer Gruppengröße von zwei - stabil ist. Diese Zeit entspricht der durchschnittlichen Dauer, die der Zusatzpunkt benötigt den Kommunikati-



a) b)

Abbildung 6.13: Stabilitätszeit von lokalen Gruppen in Abhängigkeit von der Gruppengröße für verschiedene Geschwindigkeiten

onsbereich zu durchwandern. Diese Zeitdauer $t_2^{s_\ell}$ ergibt sich durch

$$\begin{aligned} t_2^{s_\ell} &= \frac{s}{v}, \\ \text{mit } s &= \frac{r_K^2\pi}{2r_K} = \frac{1}{2}r_K\pi \\ \text{ergibt sich } t_2^{s_\ell} &= \frac{r_K\pi}{2v}. \end{aligned} \tag{6.24}$$

t_2^{sL} spiegelt die Stabilitätszeit eines Zusatzpunktes wider. Bei zwei Zusatzpunkten und somit einer Gruppengröße von drei Personen halbiert sich die Stabilitätszeit gegenüber einem Zusatzpunkt, da sich die Wahrscheinlichkeit, dass sich ein Zusatzpunkt aus dem Kommunikationsbereich bewegt, verdoppelt. Für weitere Zusatzpunkte reduziert sich die Stabilitätszeit auf ähnliche Weise, d. h.

$$t_3^{s_\ell} = \frac{1}{2} \cdot t_2^{sL},$$
$$t_4^{s_\ell} = \frac{1}{3} \cdot t_2^{sL}.$$

Für eine lokale Gruppe mit n Teilnehmern, d. h. (n-1) Zusatzpunkten folgt somit

$$t_n^{sL} = \frac{1}{n-1} \cdot t_2^{sL}, \tag{6.25}$$

so dass sich schließlich

$$t_n^{sL} = \frac{1}{n-1}\frac{r_K\pi}{2v} \tag{6.26}$$

ergibt.

Um zu verdeutlichen, wie die diese analytischen Ergebnisse mit den Simulationsergebnissen übereinstimmen, ist in Abbildung 6.14 ein Vergleich für verschiedene Geschwindigkeiten zu sehen. Hierin sind die Simulationsdaten mit Punkten in die Diagramme eingezeichnet, die Kurven verdeutlichen das analytische Ergebnis.

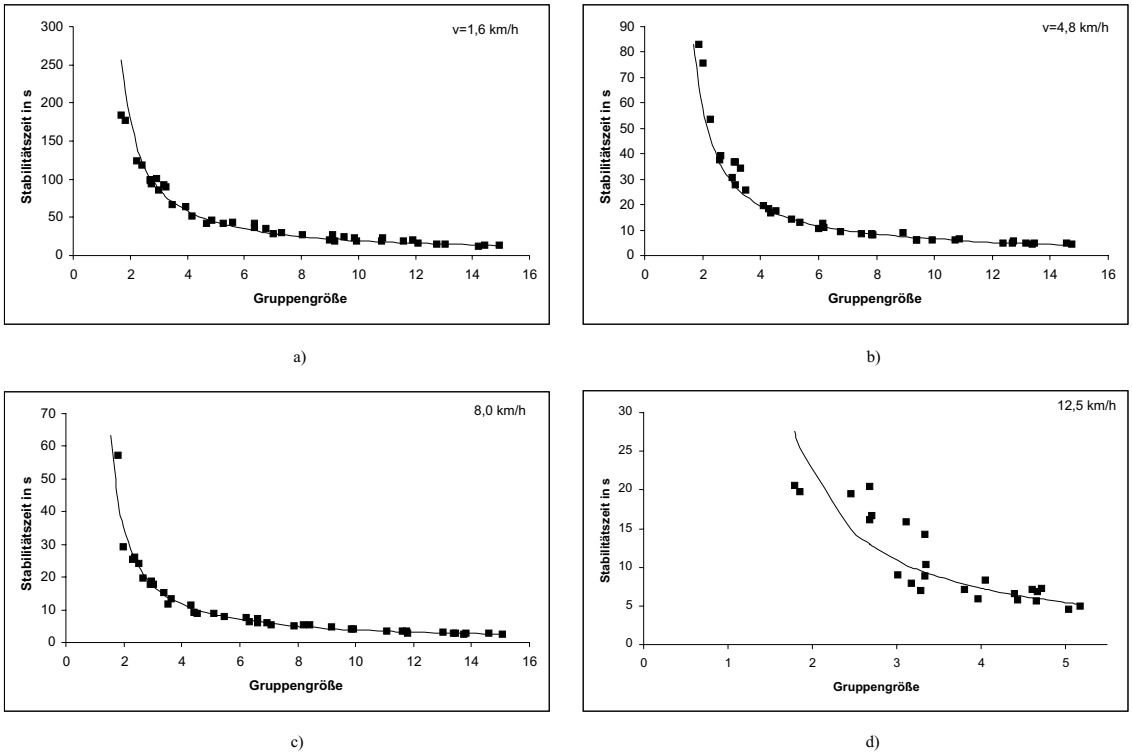


Abbildung 6.14: Gegenüberstellung der Simulationsergebnisse mit Daten, die sich durch das mathematische Modell ergeben.

Neben der Stabilitätszeit der lokalen Gruppen ist auch noch deren Größe von Interesse. Diese Größe hängt jedoch entscheidend von der verwendeten Preisbildungsvariante ab. So wird in Abschnitt 5.3.2 festgestellt, dass bei Verwendung der zweiten Variante prinzipiell keine Bedingung angegeben werden kann, dass weitere Teilnehmer nicht hinzugefügt werden können. Bei den beiden anderen Varianten ist dies nicht der Fall. Dies bestätigt auch Abbildung 6.15, welche die Größe der lokalen Gruppen für sämtliche Preisbildungsvarianten darstellt. Der Graph mit der Bezeichnung „keine Variante“ beinhaltet sämtliche direkten Kommunikationspartner und stellt deshalb für alle anderen Varianten eine obere Schranke dar. Die erste und dritte Variante liefern hinsichtlich der Gruppengröße ähnliche Ergebnisse und übersteigen eine Gruppengröße von fünf Knoten nur marginal. Die zweite Variante liegt aus den bereits diskutierten Gründen über den beiden anderen Varianten.

Bisher beschränkte sich die Untersuchung auf eine willkürliche Bewegung der Knoten. Abbildung 6.16a stellt die Stabilitätszeit t_s^G in Abhängigkeit der Gruppengröße für eine Bewegung mit Vorzugsrichtung dar. Abbildung 6.16b vergleicht die Stabilitätszeit t_s^G lokaler Gruppen bei willkürlicher Bewegung mit der Stabilitätszeit bei einer vorhandenen Vorzugsrichtung der Teilnehmer.

In beiden Fällen lässt sich erkennen, dass bei vorhandener Vorzugsrichtung die Varianz größer ist. Obwohl die Grundtendenz bei beiden Bewegungsmustern gleich ist, lässt

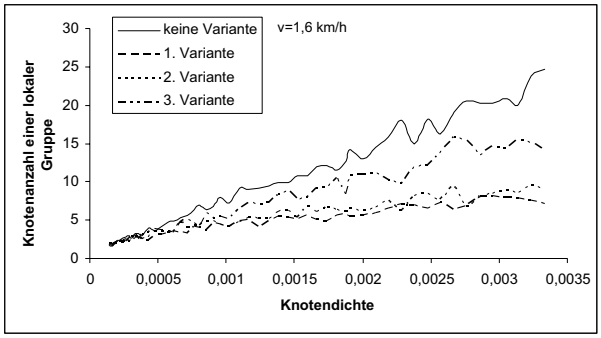
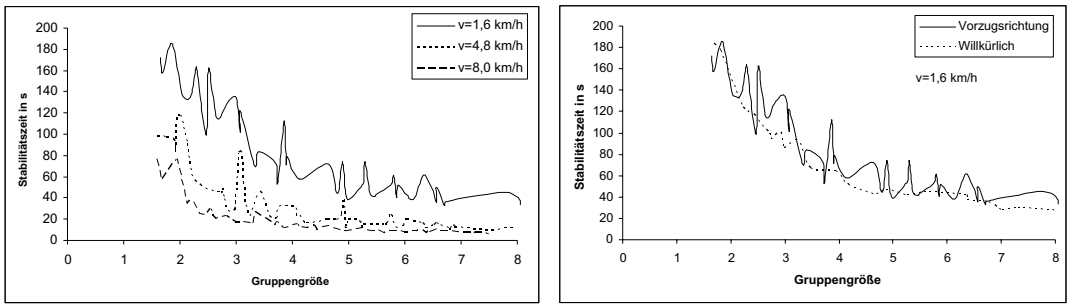


Abbildung 6.15: Größe von lokalen Gruppen für die verschiedenen Preisbildungsvarianten



a) b)

Abbildung 6.16: Vergleich der Stabilität für zufällige Bewegungsrichtungen und im Fall einer vorherrschenden Vorzugsrichtung

Abbildung 6.16b erkennen, dass bei vorliegender willkürlicher Bewegung die Abnahme der Stabilitätszeit gleichmäßiger erfolgt und auch als untere Schranke für die Bewegung mit Vorzugsrichtung angesehen wird.

6.2.3.2 Verteilte Gruppenbildung

Abschließend werden noch Simulationsergebnisse für die verteilte Gruppenbildung präsentiert. Es erfolgt ebenfalls eine Betrachtung der Stabilitätszeit sowie der Gruppengröße, wie bereits bei der lokalen Gruppenbildung.

Die Abbildungen 6.17a-6.17c zeigen die Stabilitätszeit globaler Gruppen in Abhängigkeit von der Gruppengröße und für verschiedene mittlere Geschwindigkeiten v. Die Punkte stellen die eigentlichen Messwerte dar, während die Kurve den Mittelwert darstellt. Abbildung 6.17d stellt die Mittelwertkurven für verschiedene Geschwindigkeiten in einem Diagramm dar, um auf diese Weise besser einen Vergleich durchführen zu können.

Wie in den Abbildungen 6.17a-6.17c zu sehen ist, liegt für alle untersuchten Geschwin-

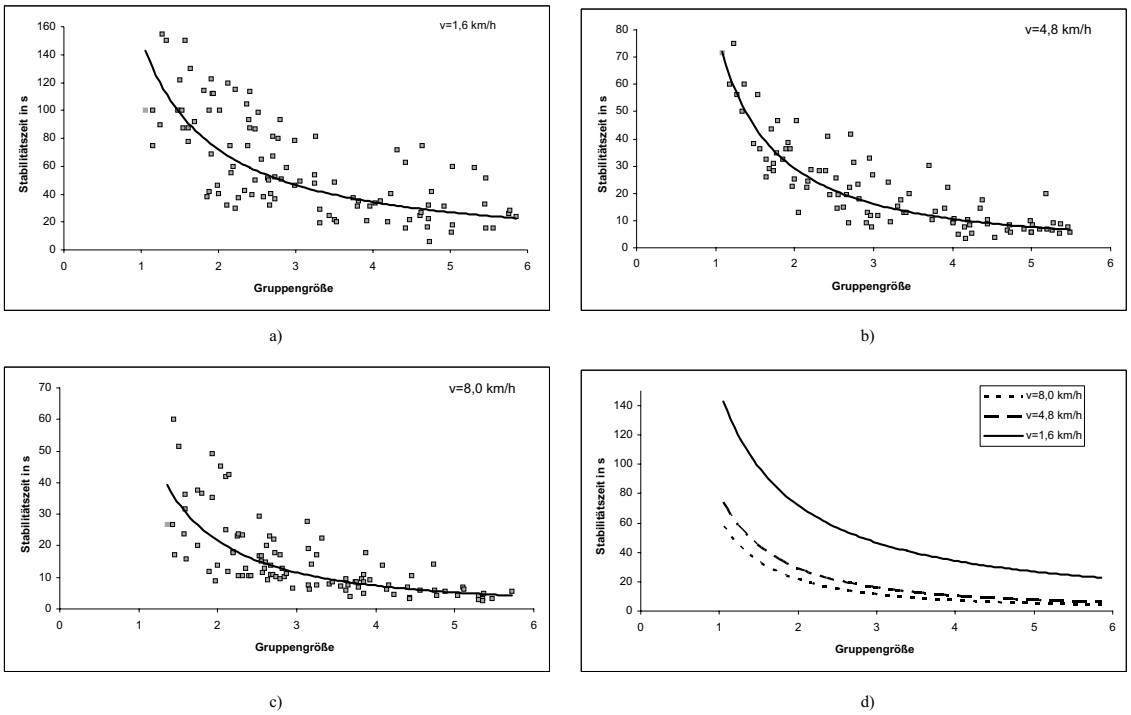


Abbildung 6.17: Stabilität von globalen Gruppen für unterschiedliche Geschwindigkeiten v.

digkeiten eine hohe Varianz vor. Diese resultiert daraus, dass in der globalen Gruppe nicht nur direkte Kommunikation innerhalb des Kommunikationsradius r_K stattfindet, sondern auch noch über mindestens einen Routerknoten erfolgt. Hierdurch kann der Kommunikationszeitraum zunehmen, da der effektive Kommunikationsradius sich erhöht. Der Kommunikationszeitraum kann aber auch abnehmen, da der Routerknoten zu jeder Zeit sich aus dem ad hoc Netzwerk entfernen kann.

Bei einer mittleren Geschwindigkeit von 8 km/h ist eine durchschnittliche Stabilitätszeit bei einer Gruppengröße von vier Personen von $5,4$ Sekunden zu verzeichnen. Jedoch beträgt die Varianz $1,8$ Sekunden, was immerhin 33 % entspricht.

Da bei der verteilten Gruppenbildung die Kommunikation auch über Routerknoten durchgeführt wird, nimmt die durchschnittliche Stabilitätszeit ab, wie Abbildung 6.18 zeigt. Wieder deuten die Punkte die Messwerte für die Stabilitätszeit der globalen Gruppe und die durchgezogene Linie deren Mittelwert an. Zu erkennen ist wieder die große Varianz. In manchen Fällen übertrifft die Stabilitätszeit einer globalen Gruppe die einer lokalen Gruppe, was auf den effektiv größeren Kommunikationsradius zurückzuführen ist. Im Durchschnitt ist jedoch die Stabilitätszeit der globalen Gruppen deutlich geringer als bei der lokalen Gruppe. Im Diagramm ist ersichtlich, dass die Stabilitätszeit der globalen Gruppe ungefähr die Hälfte der Stabilitätszeit der lokalen Gruppe beträgt.

Eine analytische Bewertung der Stabilitätszeit der globalen Gruppe ist sehr schwie-

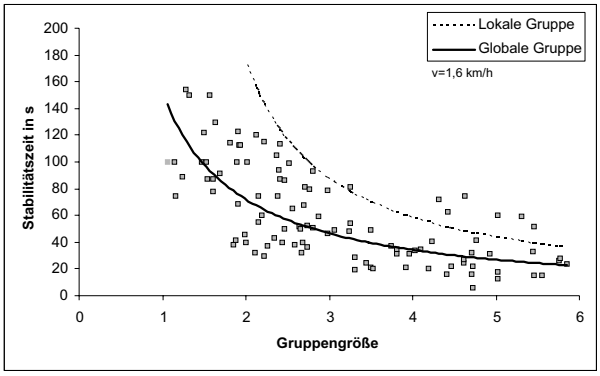


Abbildung 6.18: Vergleich der Stabilitätszeit lokaler und globaler Gruppen.

rig durchzuführen, da die Stabilitätszeit in diesem Fall von zwei relativen Geschwindigkeiten und willkürlichen Bewegungsrichtungen abhängen. Eine maximale obere Schranke lässt sich angeben, wenn der komplette Kommunikationsbereich bei 1-Hop Kommunikation durchwandert wird und stets auch ein Routerknoten zur Verfügung steht. In diesem Fall beträgt die durchschnittliche Stabilitätszeit globaler Gruppen das doppelte der Stabilitätszeit lokaler Gruppen, da der Kommunikationsbereich größer ist.

Ein Mindestmaß für die untere Grenze der Stabilitätszeit existiert nicht, so dass für deren Wert konstant Null angegeben werden muss.

Für die Angabe eines Mittelwertes wird angenommen, dass ein Knoten beim Durchwandern des halben durchschnittlichen Kommunikationsbereichs (siehe Bereich mit Radius r_{M1} in Abschnitt 6.2.1) zu 50% der Zeit eine 1-Hop Kommunikation durchführen kann.

Die durchschnittliche Stabilitätszeit $t_n^{s_G}$ in globalen Gruppen mit n Teilnehmern ist durch

$$t_n^{s_G} = \frac{s_n}{4v}$$

gegeben, wobei s_n den mittleren Weg bezeichnet, der während der Zeit zurückgelegt wird, in der die globale Gruppe stabil ist. Der Faktor vier im Nenner ergibt sich gemäß den Annahmen dadurch, dass nur zur Hälfte der Zeit eine Kommunikation stattfindet und auch nur die Hälfte des Kreises durchwandert wird.

Hinsichtlich des durchschnittlichen Kommunikationsweges für zwei Knoten s_2 gilt

$$s_2 = \frac{r_{M1}^2\pi}{2r_{M1}} = \frac{1}{2}r_{M1}\pi = \frac{\pi}{6}\sqrt{21}r_K, \tag{6.27}$$

wobei r_{M1} dem mittleren Radius bei 1-Hop-Kommunikation aus Abbildung 6.7 entspricht.

Die mittlere Stabilitätszeit bei einer Gruppengröße von zwei Knoten beläuft sich somit auf

$$t_2^{s_G} = \frac{\pi\sqrt{21}r_K}{24v}. \tag{6.28}$$

Somit würde sich bei der globalen Gruppenbildung die durchschnittliche Stabilitätszeit auf 60% gegenüber der lokalen Gruppenbildung reduzieren. Die Simulationsergebnisse spiegeln eine 50% Reduktion wider.

Bisher wurden noch keine Aussagen über die Größe globaler Gruppen präsentiert. Aus diesem Grund informiert Abbildung 6.19 über die Abhängigkeit der Größe einer globalen

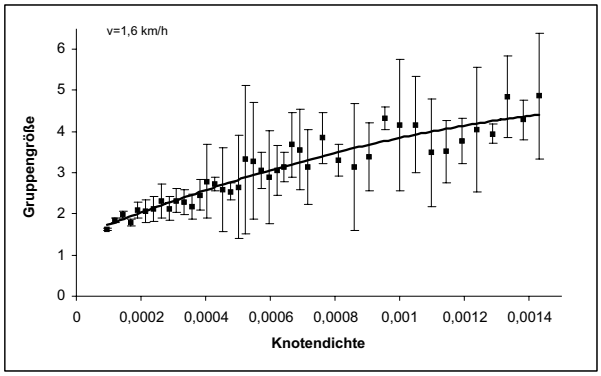


Abbildung 6.19: Abhängigkeit der Gruppengröße globaler Gruppen von der Knotendichte mit Varianzintervall.

Gruppe von der Knotendichte. Zu erkennen ist, dass bereits ab einer Knotendichte von $0,0002$ eine globale Gruppe von zwei Personen entsteht. Ab einer Knotendichte von $0,0011$ entstehen Gruppen, die die Kapazität eines normalen Taxis von vier Personen bereits übersteigen.

Zu dieser Auswertung muss jedoch hinzugefügt werden, dass die Zielkoordinaten in der Simulation uniform verteilt sind. Dies wird in der Realität i. A. nicht eintreten. So wird es in Städten Bereiche geben, die häufiger Ziel sind als andere. Aus diesem Grund wird sich die Gruppengröße an viel befahrenen Punkten erhöhen und dafür in den übrigen Bereichen verringern.

Ferner muss erwähnt werden, dass der Bereich, in dem die Zielkoordinaten liegen, beschränkt ist, d. h. sämtliche Gruppierungsteilnehmer haben eine Zielkoordinate, die sich in dem Bereich befindet. In der Realität wird dieser Zielgebietsbereich deutlich größer sein.

6.2.4 Heuristik

Abschnitt 5.4.1 beschreibt die verwendeten Heuristiken beim Taxi-Sharing-Szenario. In dem folgenden Abschnitt wird die Qualität der Heuristik gezeigt. Um die Güte der Lokales-Maximum-Heuristik darzustellen sind nachfolgend Tabellen aufgeführt, die die Heuristik bezüglich Zeitersparnis und Qualität beurteilen. Den Tabellen liegt eine Stichprobe von 50000 Simulationen zu Grunde.

In Tabelle 6.2 wird die Reduzierung der Ausführungszeit für die Gruppenbildung unter Verwendung der Heuristik verdeutlicht. Jede Zeile in Tabelle 6.2 ist Resultat einer Vielzahl an Gruppierungen. Die jeweilige Anzahl kann in der ersten Spalte gefunden werden. Die weiteren Spalten geben an, wie lange eine Gruppierung ohne und mit Heuristik andauert. Auf den ersten Blick ist zu sehen, dass mit der Lokales-Maximum-Heuristik die Gruppierungszeit erheblich reduziert wird und fast als konstant angesehen werden kann.

	mit Heuristik		ohne Heuristik	
Gruppen-größe	Zeit pro Gruppe in s	Standard-abweichung σ	Zeit pro Gruppe in s	Standard-abweichung σ
6	0,307789	0,012997	14,24903	0,816531
8	0,316586	0,014700	46,24712	3,094692

Tabelle 6.2: Absolute Zeitersparnis durch die Lokales-Maximum-Heuristik

Die Tatsache, dass ab Zeile zwölf die Gruppierungszeit ohne Heuristik mehr als verdreifacht wird, ist auf größere Gruppen zurückzuführen. Je größer die Gruppe, desto mehr Kombinationen müssen untersucht werden um die optimale Gruppe zu finden. Die Heuristik ist von diesem Anstieg nicht betroffen.

Zu erwähnen ist, dass die absoluten Zeiten in Tabelle 6.2 auf Simulationen basieren. Die Zeiten stellen direkte Ausführungszeiten des Algorithmus dar, wobei die Nachrichtenübermittlungszeit vernachlässigt werden. Für die Auswertung ist dieses Verfahren legitim, da die Nachrichtenübermittlungszeit mit und ohne Heuristik gleich ist.

Tabelle 6.3 zeigt anhand von prozentualen Daten, wie sich die Anwendung der Lokales-

gleiche Zeit	länger mit Heuristik	länger ohne Heuristik	Zeit-ersparnis
87,83 %	1,55 %	10,62 %	97,66 %

Tabelle 6.3: Relative Zeitersparnis durch die Lokales-Maximum-Heuristik

Maximum-Heuristik auswirkt. So ist in Spalte Eins zu sehen, dass in ungefähr 87 % der Gruppierungen die Heuristik keinen Geschwindigkeitsvorteil erbringt. Zeigt die zweite Spalte an, dass in 1.5 % der untersuchten Fälle die Heuristik schlechtere Ergebnisse liefert, so bedarf dies weiterer Erklärung. Zur Geschwindigkeitsmessung wurde die Systemzeit des Rechners verwendet, die eine Auflösung im Millisekundenbereich aufweist. Wenn die Heuristik mehr Zeit benötigt, so liegt der Wert immer im Millisekundenbereich, was auch auf Rundungsfehler oder Prozessumschalteverzögerung zurückzuführen sein kann.

Die dritte Spalte sagt aus, dass in mehr als 10 % der Fälle das Gruppieren mit Heuristik schneller ist als ohne. Mögen diese 10 % auf den ersten Blick als eher unterdurchschnittlicher Wert erscheinen, muss hierzu noch die vierte Spalte zur Verdeutlichung herangezogen werden. Diese Spalte gibt die Zeitersparnis wieder, die durch die Heuristik erzielt wird. Somit werden in den 10 % der Fälle, in denen die Heuristik schnellere Ergebnisse liefert, über 98 % der Zeit eingespart.

Hieraus wird deutlich, dass die Lokales-Maximum-Heuristik bei großen Gruppen zum Tragen kommt. Die 87 % bedeuten, dass es sich um kleinere Gruppen handelt und die Heuristik eigentlich nicht notwendig ist. Bei den 10 % sind die Gruppen groß, was bedeutet, dass ohne Heuristik die Gruppenbildung sehr lange (Tabelle 6.2 zeigt als Durchschnittswert in Zeile 18, Spalte 5 bereits mehr als 50 Sekunden) dauert und die Lokales-Maximum-Heuristik eine wesentlich Zeitersparnis liefert.

Die letzte Bewertungsgrundlage ist die Qualität der Ergebnisse, die mit der Heuristik erzielt wird. Es ist nicht ausreichend schnelle Ergebnisse zu liefern, wie in den obigen Tabellen zu sehen. Die Resultate dürfen auch nicht zu sehr vom Verfahren ohne Heuristik abweichen. Doch auch die Güte der Lokales-Maximum-Heuristik ist gegeben.

So zeigt Tabelle 6.4, dass sich in über 90 % der untersuchten Fälle die Gruppe, die

Totale Übereinstimmung	Keine Übereinstimmung	Pfadlänge l_P mit Heuristik	σ_{l_P}
91,26 %	3,97 %	109,88 %	0,48793

Tabelle 6.4: Qualität der Heuristik

sich durch Anwendung der Heuristik ergibt, nicht von der Gruppe unterscheidet, wenn keine Heuristik verwendet wird. Nur in im Durchschnitt weniger als 4 % sind die Teilnehmer völlig unterschiedlich. In den restlichen 5 % der Fälle sind in der Heuristik-Gruppe Teilnehmer von der nicht-Heuristik-Gruppe als auch andere vorhanden.

Tabelle 6.4 zeigt ferner, wie sich die Pfadlänge durch die Anwendung der Heuristik ändert. Da die Heuristik i. A. kein optimales Ergebnis liefert, verlängert sich die Pfadlänge. Tabelle 6.4 lässt jedoch erkennen, dass die Pfadlänge nur um durchschnittlich 10,0 % zunimmt. Zu beachten ist auch die geringe Varianz der Pfadlänge, die sich auf nur 0,220 beläuft.

Obige Ergebnisse bestätigen, dass sich die Lokales-Maximum-Heuristik sehr gut für die Taxi-Sharing-Anwendung eignet. Die Ergebnisse lassen sich auch auf andere Domänen übertragen, jedoch nur unter der Voraussetzung, dass eine ähnliche Gruppierungsfunktion verwendet wird.

6.2.5 Übertragungszeit von Nachrichten

Abschließend wird noch die Zeit untersucht, die von Geräten benötigt wird um Nachrichten auszutauschen. Tabelle 6.5 zeigt die benötigte Übertragungszeit für den Nachrichtenaustausch zwischen PDAs und Laptops. Gemessen wird die Zeit, die Gerät A benötigt eine Nachricht zu verschicken, die Gerät B benötigt um den Inhalt der Nachricht zu interpretieren (Parsing etc.) und wieder an Gerät A zurückzuschicken. Durchgeführt wurden die Messungen mit einem *Ipaq 3660* sowie mit einem 600 MHz Notebook und einhundert Mal wiederholt, um repräsentative Daten zu erhalten.

Wie in Tabelle 6.5 ersichtlich, ist die Übertragungszeit zwischen zwei PDAs für den praktischen Einsatz zu hoch. Obwohl beim Taxi-Sharing-Szenario die Stabilitätszeit für Gruppen bis zu vier Personen auch im oberen Geschwindigkeitsbereich im 4-5 Sekundenbereich liegt, besagt das Ergebnis in Tabelle 6.5, dass ein Gerät in dieser Zeit nur maximal fünf Nachrichten aussenden kann, um die Stabilitätszeit nicht zu übertreffen.

Bei der Kommunikation Laptop-Laptop ist die Kommunikationszeit auf Grund der höheren Leistungsdaten erheblich besser. Die Daten würde das Aussenden von über 40 Nachrichten innerhalb der Stabilitätszeit beim Taxi-Sharing Szenario erlauben, was i. A.

	Mittelwert Übertragungszeit in ms	Standard-abweichung in ms
PDA - PDA	865,2	53,5
Laptop - Laptop	87,26	7,57

Tabelle 6.5: Übertragungszeit von Nachrichten

für die Anwendung ausreicht.

Der Vergleich Laptop-PDA wurde deshalb gewählt, weil die nächste Generation von PDAs wohl sehr nahe an die Leistungsdaten des Laptops herankommen wird. Somit wird auch eine praktische Umsetzung hinsichtlich der Übertragungszeit in nächster Zeit möglich sein.

6.3 Zusammenfassung

Dieses Kapitel fasst die Simulationsergebnisse, die aus der Analyse des Taxi-Sharing Szenarios resultieren, zusammen mit dem Ziel, Aussagen über die Anwendbarkeit von MoPiDiG zu ermöglichen. Die Ergebnisse beziehen sich auf Verwendung von zusätzlichen virtuellen Topologien, den vorgestellten Heuristiken und auf den Gruppierungsprozess.

Im Weiteren werden Simulationsergebnisse des Taxi-Sharing Szenarios präsentiert. Untersuchungen hinsichtlich der Anzahl an Kommunikationspartnern ergeben, dass bereits bei vorliegender 1-Hop Kommunikation auch bei niedriger Knotendichte ausreichend potenzielle Gruppierungsteilnehmer zur Verfügung stehen.

Eine Analyse der virtuellen Topologie ergibt, dass diese nur für Segmente mit einer gewissen Obergrenze an Knoten einsetzbar sind. Diese Obergrenze ist abhängig von der Geschwindigkeit der Teilnehmer.

Die Stabilitätszeit - d. h. die Zeitspanne, in der kein Teilnehmer die Gruppe verlässt bzw. ein neuer Teilnehmer hinzukommt - beläuft sich für die Maximalgruppenanzahl im Taxi-Sharing-Szenario auf den vier-Sekunden Bereich.

Die verwendeten Heuristiken werden ebenfalls untersucht und die Ergebnisse bestätigen, dass mit Hilfe dieser Heuristiken die Ausführzeiten erheblich reduziert werden und innerhalb der Stabilitätszeit erhalten werden können. Die Qualität des Ergebnisses wird nur geringfügig beeinträchtigt.

Hinsichtlich der Kommunikationszeit ist festzustellen, dass der Nachrichtenaustausch zweier PDAs zu viel Zeit in Anspruch nimmt um in der Praxis zu bestehen zu können. Der Vergleich mit einem System, welches hinsichtlich der Leistung der nächsten PDA-Generation sehr nahe kommt, lässt jedoch eine rasche mögliche praktische Umsetzung vermuten.

Kapitel 7

Zusammenfassung

Im Mittelpunkt dieser Arbeit steht die Entwicklung des Frameworks MoPiDiG *Mobile Profile based Distributed Grouping* zur profilbasierten Gruppenbildung in ad hoc Netzwerken. Mit MoPiDiG ist es möglich ohne zusätzliche Infrastruktur Personen auf Basis von Profildaten zu Gruppen zusammenzufassen.

Im MoPiDiG Framework besitzt jeder Benutzer ein mobiles Endgerät, auf welchem sich sein Benutzerprofil befindet. Die Kommunikation zwischen den Endgeräten findet mit so genannten spontanen (ad hoc) Netzwerken statt, die sich dann bilden, wenn sich Geräte in Kommunikationsreichweite befinden. Innerhalb der Kommunikationspartner versucht MoPiDiG Gruppen zu finden. Eine Gruppe zeichnet sich dadurch aus, dass deren Mitglieder ähnliche Profile aufweisen [BS03].

Das MoPiDiG Framework ist auf keine spezielle Domäne begrenzt. Ferner wird zur Gruppenbildung keine Ortsinformation oder Geo-Koordinate benötigt und es ist kein zentraler Koordinator notwendig.

In MoPiDiG werden zwei Arten von Gruppen betrachtet. Im ersten Fall schließt sich eine Anzahl an Objekten (z. B. Menschen oder Maschinen) zu einer Gruppe zusammen, da sie einzeln nicht in der Lage sind ein gewisses Ziel zu erreichen. Die Gruppe ist somit Voraussetzung um das Ziel zu erreichen. Diese Art der Gruppenbildung ist die ähnlichkeitsbasierte Gruppenbildung.

Im anderen Fall kann von jedem Objekt das Ziel auch allein erreicht werden. Durch Zusammenschluss zu einer Gruppe kann das Ziel jedoch schneller, besser oder mit weniger Aufwand erreicht werden. Dieser Prozess wird als nutzenbasierte Gruppenbildung bezeichnet.

Die Architektur des MoPiDiG Frameworks ist modular und besteht aus mehreren Schichten. Das MoPiDiG Framework setzt als Basis eine ad hoc Middleware voraus, welche primär für das Senden und Empfangen von Nachrichten verantwortlich ist.

Über dieser Middleware ist der eigentliche domänenunabhängige Kern des MoPiDiG Frameworks angesiedelt. Er besteht zum einem aus einer Ansammlung von algorithmischen Verfahren, die zur Gruppenbildung notwendig sind und zum anderen aus Meta-Beschreibungen für Profil- und Gruppenbeschreibung, die die Brücke zur darüber liegenden domänenabhängigen Schicht bilden. In dieser Schicht kann eine MoPiDiG Anwendung den Gegebenheiten angepasst werden. So werden hier Profile- und Gruppenbeschreibungen angegeben. Ferner existiert noch eine Domänenbeschreibung, in der die Anwendung

konfiguriert werden kann und außerdem noch Schnittstellen zur domänenunabhängigen Schicht existieren [SB03; SBB04a].

Bezüglich der Algorithmen reicht es nicht aus, sich nur auf die Gruppierung zu konzentrieren [MSUB04; SBB04b; SBB05]. Da ad hoc Netzwerke aus einer großen Anzahl an Teilnehmern bestehen können, muss eine Methode vorhanden sein, die ein großes Netzwerk in mehrere kleinere aufteilt. Der Parameter, der festlegt, wie groß das Segment werden wird, wird durch den Durchmesser des Segments spezifiziert.

Damit diese Segmentierung nicht von allen Teilnehmern gleichzeitig vollzogen wird, ist eine Initiatorbestimmung notwendig, die die initiierenden Teilnehmer festlegt. Hier sind in der Literatur aktive und passive Verfahren bekannt, wobei im MoPiDiG Framework nur passive Methoden zum Einsatz kommen, da diese auf das Senden von Nachrichten verzichten und den Initiator implizit festlegen.

Um die Anzahl an Nachrichten zu reduzieren, kann eine virtuelle Topologie kreiert werden, sofern es das Geschwindigkeitsprofil erlaubt. In der vorliegenden Arbeit wurden Verfahren beschrieben, wie eine virtuelle Baum- und Ringtopologie in das Gruppierungsverfahren mit einbezogen wird. Dies schließt sowohl die Erzeugung der entsprechenden Strukturen als auch die Aufrechterhaltung mit ein.

Die Gruppierung ist der zentrale Punkt bei MoPiDiG. Der Gruppierungsprozess wird in zwei Schritten vollzogen. Im ersten Schritt wird eine lokale Gruppe erzeugt, die aus einer Teilmenge der direkten Nachbarn besteht. Diese lokalen Gruppen werden im zweiten Schritt ausgetauscht, bis schließlich eine globale Gruppe vorhanden ist.

Mit dem MoPiDiG Framework ist es möglich sowohl ähnlichkeits- als auch nutzenbasierte Gruppen zu bilden. Für das ähnlichkeitsbasierte Gruppieren ist eine Distanzfunktion notwendig, die entscheidet, ob Profile ähnlich sind. Diese Distanzfunktion kann von Anwendung zu Anwendung unterschiedlich sein und wird deshalb im domänenabhängigen Teil von MoPiDiG definiert. Für die nutzenbasierte Gruppierung ist eine Nutzenfunktion notwendig. Eine Menge an Personen schließt sich nur zu einer Gruppe zusammen, wenn die Nutzenfunktion jedes einzelnen dadurch erhöht wird. Die Nutzenfunktion wird ebenfalls im domänenabhängigen Teil von MoPiDiG definiert.

Beide Gruppierungsarten haben den Charakter einer Optimierungsaufgabe inne. Beim ähnlichkeitsbasierten Gruppieren müssen aus einer Menge an Personen die ähnlichsten gefunden werden und beim nutzenbasierten Gruppieren muss die Nutzenfunktion maximiert werden. Da jedoch eine MoPiDiG Anwendung im mobilen Umfeld ausgeführt wird, besteht häufig nicht die Zeit, diese Optimierungsaufgabe so zu lösen, dass ein globales Optimum erreicht wird. Aus diesem Grund wurden für sämtliche Gruppierungsarten Heuristiken präsentiert, so dass zum einem das Ergebnis nicht zu stark in Mitleidenschaft gezogen wird, und zum anderen die Ausführungszeit extrem reduziert wird.

Neben dem allgemeinen MoPiDiG Framework wird in dieser Arbeit noch eine konkrete Anwendung beschrieben. Hierbei handelt es sich um ein Taxi-Sharing Szenario. Ausgangspunkt ist hierfür ein Bahnhof oder Flughafengebäude, in dem sich Personen befinden, die mit Hilfe eines Taxis an ihr endgültiges Ziel gelangen wollen.

Die Teilnehmer tauschen ihre Profilinformationen aus, um auf diese Weise eine oder mehrere Gruppen zu erzeugen. Zu einer Gruppe schließen sich Teilnehmer zusammen, wenn sie entweder das gleiche Fahrtziel haben, oder das Ziel ohne große Umwege zu erreichen ist. Grundvoraussetzung für die Gruppenbildung ist, dass sich der Fahrpreis für jede

teilnehmende Person im Vergleich zur Solofahrt reduziert [SB03].

Da für die praktische Umsetzung eine Vielzahl an Ressourcen notwendig sind, wurde eine Simulationsumgebung entwickelt, mit der für das Taxi-Sharing Szenario sämtliche Verfahren und Algorithmen analysiert wurden. Die Ergebnisse, die auf diese Weise erhalten wurden, sind im nächsten Abschnitt wiedergegeben.

7.1 Ergebnisse

Die Architektur und die verwendeten Algorithmen wurden am Beispiel des Taxi-Sharing Szenarios intensiv untersucht.

Mit der Simulationsumgebung ist es prinzipiell möglich Untersuchungen für eine beliebige Anzahl an Routerknoten (Hop-Anzahl) durchzuführen. Analysen bezüglich der Segmentgröße ergeben jedoch, dass bereits bei vorhandener 1-Hop-Kommunikation und geringen Knotendichten die Segmentgröße ausreichend ist, um eine genügende Anzahl an Kommunikationspartnern zur Verfügung zu haben. Größere Segmente bedeuten zunehmende Instabilität und eine steigende Anzahl an Nachrichten, die notwendig sind, die Segmente aufrechtzuerhalten.

Hinsichtlich der Stabilitätszeit, d. h. die Zeit, in der kein Teilnehmer der Gruppe diese verlässt, ist folgender Zusammenhang für die Stabilitätszeit t_n^{sL} für lokale Gruppen und t_n^{sG} für globale Gruppen mit jeweils n Teilnehmer festzustellen:

$$t_n^{sL} = \frac{1}{n-1}\frac{r_K \pi}{2v}, \tag{7.1}$$

$$t_n^{sG} = \frac{1}{n-1}\frac{\pi\sqrt{21} r_K}{24v}. \tag{7.2}$$

Für eine globale Gruppe mit vier Personen beträgt die Stabilitätszeit auch im oberen Geschwindigkeitsbereich ($\sim$ 8 km/h) 4-5 Sekunden.

Diese Zeit ist ausschlaggebend, da - wenn die Gruppierung innerhalb dieser Zeitspanne abgeschlossen ist - mit großer Wahrscheinlichkeit sichergestellt ist, dass sämtliche zukünftigen Gruppenmitglieder über ihre Gruppenzugehörigkeit informiert sind.

Untersuchungen der Gruppierungszeit haben ergeben, dass durchschnittlich 14 Sekunden benötigt werden, eine Gruppe mit sechs Teilnehmern zu bilden, wenn ein globales Optimum erreicht werden soll. Dieser Wert steigt auf 46 Sekunden, wenn eine Gruppengröße von acht Personen berechnet werden sollen. Diese Zeit bezieht sich auf rein auf den Algorithmus, eine Kommunikationszeit ist nicht berücksichtigt. Diese Gruppenerzeugungszeiten sind für den praktischen Einsatz zu hoch.

Werden die Heuristiken angewandt, reduziert sich die Gruppenerzeugungszeit auf 300 Millisekunden für die Gruppe mit sechs Personen und auch die Gruppe der Größe acht liegt mit 320 Millisekunden nur unerheblich darüber. Diese Werte sprechen für die Anwendbarkeit der Heuristik.

Durch die Anwendung der Heuristiken wird der Suchraum für die Optimierung der Profitfunktion begrenzt. Deswegen wird kein globales Optimum erzeugt, sondern nur ein lokales. Der durchschnittliche Qualitätsverlust bei Verwendung der Heuristik beträgt jedoch nur 10%. Dieser Wert ist hinsichtlich der erbrachten Zeitersparnis als sehr akzeptabel anzusehen.

Negativ wirkt sich jedoch die Kommunikationszeit bei vorliegender PDA zu PDA WLAN-Kommunikation aus. So dauert das Aussenden einer Nachricht bis zum Erhalt einer Empfangsbestätigung mit der verwendeten Middleware und zweier IPaQs 3660 durchschnittlich 0,87 Sekunden. Dieser Wert ist für die Praxis zu hoch. Die Entwicklung neuerer leistungsfähigerer PDAs geht jedoch in solch einem Maße voran, dass eine praktische Umsetzung hinsichtlich der Übertragungszeit in nächster Zeit möglich sein wird.

7.2 Ausblick

Die in dieser Arbeit vorgestellten Verfahren zur profilbasierten Gruppenbildung in ad hoc Netzwerken können in mancher Hinsicht geändert und erweitert werden.

Die zu verwendende virtuelle Topologie ist von der Anwendung abhängig. Somit wird es MoPiDiG Anwendungen geben, für die die vorgestellten Baum- und Ringtopologien nicht adäquat sind. Sollte die Notwendigkeit bestehen, neue Topologien zu MoPiDiG hinzuzufügen, so ist dies durchaus möglich. Diese neuen Topologien müssten als neue Programmmodule zu MoPiDiG hinzugefügt werden.

Ein mögliche Ergänzung wäre, sich nicht auf eine virtuelle Topologie zu beschränken, sondern mehrere gleichzeitig aufzubauen, denn Redundanz bringt Alternativen! Liegt beispielsweise ein vollvermaschtes Netzwerk vor, so können zwei Ringe erstellt werden, wobei ein Ring der Komplementärgraph[1] des anderen ist. Die Idee ist durch einen marginalen Mehraufwand mehrere Topologien zu erstellen. Treten Änderungen im Segment auf, so kann eine Topologie von diesen Änderungen nicht betroffen sein. Diese Topologie wird ab diesem Zeitpunkt zur einzig verwendeten, die restliche Topologien werden verworfen.

Es werden demzufolge mehrere Topologien erstellt, aber nur diejenige aufrechterhalten, die sich zu Beginn am günstigsten verhalten hat. Inwiefern sich dieser Ansatz positiv auswirkt und eine Leistungssteigerung erbringt, müsste mit Hilfe von Tests untersucht werden.

Gemäß den Annahmen aus der Aufgabenbeschreibung wird für das MoPiDiG Framework auf Ortsinformationen (z. B. Global Positioning System [GI00]) verzichtet und die bisherigen Verfahren zeigen, dass sie auch nicht nötig sind. Nichtsdestoweniger wären diese Daten für den Gruppierungsprozess dienlich. Mit Hilfe von Ortsinformationen wären auch Geschwindigkeitsdaten der potenziellen Gruppenmitglieder vorhanden. Für die Gruppenbildung bedeutet dies, dass nur Kommunikationspartner mit entsprechender Geschwindigkeit sich am Gruppierungsprozess beteiligen. Auf diese Weise kann die Anzahl der Nachrichten, die notwendig sind, Gruppen aufrechtzuerhalten reduziert werden und somit können größere Gruppen gebildet werden.

Bei der Existenz einer Ortsinformation kann zusätzlich auch ein Treffpunkt angegeben werden, an dem die Gruppenmitglieder zusammenkommen können. Auf eine manuelle Kontaktaufnahme, wie sie in dieser Arbeit beschrieben ist, kann somit verzichtet werden.

Bisher ist es nur möglich, eine Gruppierung gleichzeitig durchzuführen. Prinzipiell wäre es zwar durchführbar mehrere Gruppierungen parallel zu betreiben, es müsste jedoch

[1] Der Komplementärgraph $\overline{G}$ von G ist der Graph auf V, in dem zwei Knoten genau dann benachbart sind, wenn sie es in G nicht sind [Die00].

gewährleistet sein, dass die unterschiedlichen Gruppierungen auf unterschiedlichen Profilbereichen basieren. Solch ein Mechanismus könnte implementiert werden, indem die verschiedenen Gruppendefinitionen auf Disjunktheit untersucht werden und eine neue Gruppenbildung nur erlaubt wird, wenn verschiedene Profilbereiche von den Gruppierungen betroffen sind.

Eine Anwendbarkeit auf Mobiltelefone ist erwünscht und das eigentliche Ziel. Probleme mit diesen Geräten ist in erster Hinsicht die zur Zeit kaum existierende Middleware. Als Funktechnologie ist bei Funktelefonen momentan nur Bluetooth möglich. Große Bluetooth-Netzwerke (Scatternet) sind jedoch noch aktiver Forschungsgegenstand, so dass dieses Gebiet noch einen offenen Punkt der Umsetzung darstellt.

Die rasche Entwicklung auf diesen Gebieten lässt jedoch eine praktische Umsetzung in nächster Zeit erwarten.

Anhang A

Meta-Beschreibungen

Nachfolgend ist sowohl die vollständige Meta-Profilbeschreibung als auch die Meta-Gruppenbeschreibung in XML-Schema aufgelistet.

A.1 Meta-Profilbeschreibung

In der XML-Schema Meta-Profilbeschreibung wird zuerst das Tag `PROFILEENTRIES` definiert, welches sich rekursiv selbst enthält. Damit keine endlosen Kaskaden von `PROFILEENTRIES` erlaubt sind, ist der Eintrag `SUBENTRIES` optional, was mit dem Zusatzfeld `minOccurs` erreicht wird.

Danach beginnt mit der Definition des Tags `USERPROFILE` die eigentliche Profildefinition, wie in Abschnitt 4.4.1.1 erläutert wird.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

<xs:element name="PROFILEENTRIES">
<xs:complexType>
  <xs:sequence>
    <xs:element name="PROFILEENTRY" maxOccurs="unbounded">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ENTRYNAME" type="xs:string"/>
          <xs:choice>
            <xs:element name="ENTRYSTRINGVALUE" type="xs:string"/>
            <xs:element name="ENTRYNUMBERVALUE" type="xs:decimal"/>
          </xs:choice>
        <xs:element name="SUBENTRIES" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="SUBENTRYOF" type="xs:string"/>
            <xs:element ref="PROFILEENTRIES"/>
          </xs:sequence>
        </xs:complexType>
        </xs:element>
```

```
      </xs:sequence>
    </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="USERPROFILE">
<xs:complexType>
  <xs:sequence>

    <xs:element name="USERDATA">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="USERID" type="xs:string"/>
        <xs:element name="USERNAME" type="xs:string"/>
        <xs:any minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
    </xs:element>
    <xs:element ref="PROFILEENTRIES"/>

  </xs:sequence>
</xs:complexType>
</xs:element>

</xs:schema>
```

A.2 Meta-Gruppenbeschreibung

Das XML Schema der Meta-Gruppenbeschreibung beginnt mit der Definition der drei Schema-Gruppen für die jeweilige Gruppierungsart. Die eigentliche Meta-Beschreibung beginnt mit der Definition des Tags `GROUPDESCRIPTION`. Diese beinhaltet eine der drei definierten Schema-Gruppen, sowie die Angabe über die Gruppengröße und die zeitliche Gültigkeit der Gruppe.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

<xs:group name="profitbasedGroup">
  <xs:sequence>
    <xs:element name="GROUPTYPE">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="profit"/>
      </xs:restriction>
    </xs:simpleType>
    </xs:element>
```

```
    <xs:element name="ENTRIES">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ENTRYNAME" type="xs:string"
                                     maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:group>

<xs:group name="explizitsimilarGroup">
  <xs:sequence>
    <xs:element name="GROUPTYPE">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="explizit similar"/>
      </xs:restriction>
    </xs:simpleType>
    </xs:element>

    <xs:element name="ENTRIES">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PROFILEENTRIES" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="ENTRY" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="ENTRYNAME" type="xs:string"/>
                <xs:choice>
                  <xs:element name="ENTRYSTRINGVALUE" type="xs:string"/>
                  <xs:element name="ENTRYNUMBERVALUE" type="xs:decimal"/>
                </xs:choice>
              </xs:sequence>
            </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:group>

<xs:group name="implizitsimilarGroup">
  <xs:sequence>
    <xs:element name="GROUPTYPE">
    <xs:simpleType>
```

```
      <xs:restriction base="xs:string">
        <xs:enumeration value="implizit similar"/>
      </xs:restriction>
    </xs:simpleType>
    </xs:element>

    <xs:element name="ENTRIES" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ENTRYNAME" type="xs:string"
                                     maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:group>

<xs:element name="GROUPDESCRIPTION">
<xs:complexType>
  <xs:sequence>
    <xs:choice>
      <xs:group ref="profitbasedGroup"/>
      <xs:group ref="explizitsimilarGroup"/>
      <xs:group ref="implizitsimilarGroup"/>
    </xs:choice>

    <xs:element name="GROUPSIZE">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="LOWERLIMIT" type="xs:nonNegativeInteger"/>
        <xs:element name="UPPERLIMIT" type="xs:nonNegativeInteger"/>
      </xs:sequence>
    </xs:complexType>
    </xs:element>

    <xs:element name="VALIDITYPERIOD">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="START" type="xs:dateTime"/>
        <xs:element name="END" type="xs:dateTime"/>
      </xs:sequence>
    </xs:complexType>
    </xs:element>

  </xs:sequence>
</xs:complexType>
</xs:element>

</xs:schema>
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[AAL94] Hosame Abu-Amara, Jahnavi Lokre. Election in asynchronous complete networks with intermittent link failures. *IEEE Transactions on Computers*, 43(7):778–788, 1994.

[Agg01] Charu C. Aggarwal. A human-computer cooperative system for effective high dimensional clustering. In *Knowledge Discovery and Data Mining*, pages 221–226, 2001.

[AP91] Douglas E. Appelt, Martha E. Pollack. Weighted abduction for plan ascription. *User Modeling and User-Adapted Interaction*, 2(1-2):1–25, 1991.

[Ast02] Jens Astor. MOTOMM - Mobile one-to-one matchmaking. Interner Bericht, Siemens AG, 2002.

[AT01] Gediminas Adomavicius, Alexander Tuzhilin. Using data mining methods to build customer profiles. *Computer*, 34(2):74–82, 2001.

[AUN01] Thomas C. Agoston, Tatsuro Ueda, Yukari Nishimura. Pervasive computing in a networked world. In *Global Distributed Intelligence for Everyone, INET2000: 10th Annual Internet Society Conference*, Yokohama, Japan, July 2001.

[AWY+99] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, Jong Soo Park. Fast algorithms for projected clustering. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 61–72, 1999.

[BAI94] B. R. Badrinath, Arup Acharya, Tomasz Imielinski. Structuring distributed algorithms for mobile hosts. In *14th International Conference on Distributed Computing Systems*, pages 21–28, Juni 1994.

[Bas99] S. Basagni. Distributed clustering for ad hoc networks. In *Proceedings of the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, pages 310–315, 1999.

[BBH97] Jean Bacon, John Bates, David Halls. Location oriented multimedia. *IEEE Personal Communications*, 4(5):48–57, October 1997.

[BEG02] Siegfried Berninghaus, Karl-Martin Ehrhard, Werner Güth. *Strategische Spiele - Eine Einführung in die Spieltheorie*. Springer, 2002.

[Ber04] Michael Berger. Co-operating Objects – Steps Towards Ambient Intelligence. ftp://ftp.cordis.lu/pub/ist/docs/dir_c/ems/berger_en.pdf, 2004.

[BHM97] N. Badache, M. Hurfun, R. Macedo. Solving the consensus problem in a mobile environment. Technical report, IRISA, Rennes, 1997.

[BK00] Suman Banerjee, Samir Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. Technical report, Univiversity of Maryland, 2000.

[Blu01a] Bluetooth SIG. *Specification of the Bluetooth System, Specification Volume 1, Core, Version 1.1*, February 2001.

[Blu01b] Bluetooth SIG. *Specification of the Bluetooth System, Specification Volume 2, Profiles, Version 1.1*, February 2001.

[BM01] Paul V. Biron, Ashok Malhotra. *W3C Recommendation - XML Schema Part 2: Datatypes*, 2001.

[BMS05] Michael Berger, Jörg P. Müller, Christian Seitz. Multiagenten-technologien für ambient intelligence. *it – Information Technology*, 47(1):13–19, 2005.

[Bol85] Béla Bollobás. *Random Graphs*. Harcourt Brace Jovanovich Publishers, 1985.

[BPMY04] Tim Bray, Jean Paoli, Eve Maler, François Yergeau. *W3C Recommendation - Extensible Markup Language (XML) 1.0 (Third Edition)*, 2004.

[BPSM+04] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan. *W3C Recommendation - Extensible Markup Language (XML) 1.1*, 2004.

[BR93] Jeffrey D. Banfield, Adrian E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49:803–821, 1993.

[Bra94] Andreas Brandstädt. *Graphen und Algorithmen*. Teubner, Stuttgart, 1994.

[Bro96] Brockhaus GmbH. *Brockhaus Enzyklopädie*. F. A. Brockhaus GmbH, Mannheim, 20. edition, 1996.

[BS03] M. Berger, C. Seitz. Verfahren und Einrichtung sowie Computerprogramm mit Programmcode-Mitteln und Computerprogramm-Produkt zum Clustern einer Mehrzahl an Teilnehmern in einem mobilen Netzwerk. Patentanmeldung, AZ: 2003P07514 DE, August 2003.

[BSM00] Ilja N. Bronstein, Konstantin A. Semendjajew, Gerhard Musiol. *Taschenbuch der Mathematik*. Harri Deutsch, 2000.

[BT03] Urban Bastert, Gerd Thiedemann. Bluetooth - Universelle Funktechnologie zur Kommunikation. `http://www.avm.de`, Oktober 2003.

[CD99] James Clark, Steve DeRose. *W3C Recommendation - XML Path Language (XPath) Version 1.0*, 1999.

[Cha82] Ernest J. H. Chang. Echo algorithms: Depth parallel operations on general graphs. *IEEE Transactions on Software Engineering*, SE-8(4):391–401, July 1982.

[Che00] Stuart Cheshire. *Dynamic Configuration of IPv4 Link-local Addresses*. Internet Engineering Task Force, November 2000.

[CK00] Guanling Chen, David Kotz. A survey of context-aware mobile computing research. Technical Resport TR2000-381, Deptartment of Computer Science, Dartmouth College, November 2000.

[CLRS01] Thomas. H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 2001.

[CN99] Shigang Chen, Klara Nahrstedt. Distributed quality-of-service routing in ad-hoc networks. *IEEE Journal on Special Areas in Communication*, 17(8):1–18, August 1999.

[CNS00] Shigang Chen, Klara Nahrstedt, Yuval Shavitt. A QoS-aware multicast routing protocol. *IEEE Journal on Special Areas in Communication*, 2000.

[CS98] Barry Crabtree, Stuart Soltysiak. Identifying and tracking changing interests. *Int. Journal on Digital Libraries*, 2(1):38–53, 1998.

[CS99] Geng Chen, Ivan Stojmenovic. Clustering and routing in mobile wireless networks. Technical report, School of Information Technology & Engineering, University of Ottawa, Canada, June 1999.

[Cun04] Cunningham & Cunningham. Pervasive computing. `http://c2.com/cgi/-wiki?PervasiveComputing`, 2004.

[DBS+01] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, J.-C. Burgelman. Scenarios for ambient intelligence in 2010. Istag report, Information Society Technologies Advisory Group (ISTAG), February 2001.

[DCMF99] Nigel Davies, Keith Cheverst, Keith Mitchell, Adrian Friday. 'caches in the air': Disseminationg tourist information in the GUIDE system. In *Proceedings of the Second Workshop on Mobile Computing Systems and Applications, Louisiana, New Orleans*. IEEE Computer Society Press, February 1999.

[Dey01] Anind K. Dey. Understanding and using context. *Personal and Ubiquitous Computing - Special Issue on Situated Interaction and Ubiquitous Computing*, 5(1):4–7, 2001.

[Die00] Reinhard Diestel. *Graphentheorie*. Springer-Verlag Heidelberg, 2000.

[DLP+86] Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, William E. Weihl. Reaching approximate agreement in the presence of faults. *Journal of the Association for Computing Machinery*, 33(3):499–516, July 1986.

[Dro97] R. Droms. *RFCD 2131 - Dynamic Host Configuration Protocol*. Internet Engineering Task Force, March 1997.

[Eik04] Heinz-Josef Eikerling. Was ist Ambient Intelligent (AmI)? C-LAB Short Report 1, Cooperative Computing & Communication Laboratory (C-LAB), C-LAB, Fürstenallee 11, 33102 Paderborn, 2004.

[EJY04] Robert B. Ellis, Xingde Jia, Catherine Yan. On random points in the unit disk. preprint, Texas A&M University und Texas State University, February 2004.

[ES00] Martin Ester, Jörg Sander. *Knowledge Discovery in Databases - Technik und Anwendungen*. Springer, 2000.

[EWHK+96] T. Emden-Weinert, S. Hougardy, B. Kreuter, H. J. Prömel, A. Steger. *Einführung in Graphen und Algorithmen*. `http://www.informatik.-hu-berlin.de/Institut/struktur/algorithmen/ga`, 1996.

[Fal01] David C. Fallside. *W3C Recommendation - XML Schema Part 0: Primer*, 2001.

[Fas99] D. Fasulo. An analysis of recent work on clustering algorithms. Technical report, University of Washington, 1999.

[FdC03] M. Friedewald, O. da Costa. Science and technology roadmapping: Ambient intelligence in everyday life. Working paper, Fraunhofer Society, Institute for Systems and Innovation Research, Karlsruhe, July 2003.

[FIP03] FIPA. Foundations for Intelligent Physical Agents. `http://www.fipa.org`, 2003.

[FLP85] Michael J. Fischer, Nancy A. Lynch, Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the Association for Computing Machinery*, 42(2):374–382, April 1985.

[GB81] Eli M. Gafni, Dimitri P. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, COM-29(1):11–18, January 1981.

[Gel99] Hans-W. Gellersen. *Handheld and Ubiquitous Computing*. Springer, 1999.

[GI00] Inc. GARMIN International. GPS-Guide for beginners. `http://www.garmin.com`, Dezember 2000.

[Gie03] M. Gier. Wireless LAN Industrie Standards und Normen. `http://www.-abcdata.de/Wireless%20Lan%20Standards.pdf`, August 2003.

[GM82] Hector Garcia-Molina. Elections in a distributed computing system. *IEEE Transactions on Computers*, C-31(1):48–59, January 1982.

[GMP02] Laura Galluccio, Giacomo Morabito, Sergio Palazzo. Spontaneous group management in mobile ad hoc networks. In *Proceedings of Med-Hoc-Net, Sardegna, Italien*, September 2002.

[Gon01] Li Gong. JXTA: A network programming environment. *IEEE Internet Computing*, 5(3):88–95, 2001.

[Gro03] Georg Groh. Modelling ad-hoc-groups in communities. In *International Workshop Virtual Communities & Mobility, TU München*, Juni 2003.

[GT95] Mario Gerla, Jack T.-C. Tsai. Multicluster, mobile, multimedia radio network. *ACM/Baltzer Journal of Wireless Networks*, 1(3):255–265, 1995.

[GXH03] Mario Gerla, Kaixhin Xu, Xiaoyan Hong. Exploiting mobility in large scale ad hoc wireless networks. In *Proceedings of the IEEE Computer Communications Workshop 2003 (CCW 2003), Dana Point, California, USA*, Oktober 2003.

[Har75] John Hartigan. *Clustering Algorithms*. John Wiley & Sons, 1975.

[HI93] M. Holler, G. Illing. *Einführung in die Spieltheorie*. Springer, 1993.

[HKLMh03] Andreas Heinemann, Jussi Kangasharju, Fernando Lyardet, Max Mühlhäuser. iClouds - peer-to-peer information sharing in mobile environments. In Harald Kosch, László Böszörményi, Hermann Hellwagner, editors, *Euro-Par 2003. Parallel Processing, 9th International Euro-Par Conference, Klagenfurt, Austria, August 26-29, 2003. Proceedings*, volume 2790 of *Lecture Notes in Computer Science*. Springer, 2003.

[HKS+03] Lorenz Hilty, Andreas Köhler, Claudia Som, Arend Brunink, Siegfried Behrendt, Lorenz Erdmann, Felix Würtenberger, Mathias Binswanger, Niels Kuster, Jürg Fröhlich. Unser Alltag im Netz der schlauen Gegenstände. Kurzfassung TA-SWISS Studie: Das Vorsorgeprinzip in der Informationsgesellschaft. Auswirkungen des Pervasive Computing auf Gesundheit und Umwelt (TA 46A/2003), Zentrum für Technologiefolgen-Abschätzung, Zentrum für Technologiefolgen-Abschätzung, Birkenweg 61, CH-3003 Bern, 2003.

[HMNS03] Uwe Hansmann, Lothar Merk, Martin S. Nicklous, Thomas Stober. *Pervasive Computing Handbook*. Springer Berlin, 2nd edition edition, 2003.

[HPS+99] Kostas P. Hatzis, George P. Pentaris, Paul G. Spirakis, Vasilis T. Tampakas, Richard B. Tan. Fundamental control algorithms in mobile networks. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 251–260, 1999.

[IBM02] IBM. IBM Pervasive Computing Homepage. `http://www.ibm.com/pvc/-pervasive.shtml`, 2002.

[Int04] Internet Software Consortium. Internet Domain Survey. `http://www.isc.-org/index.pl?/ops/ds/reports/2004-01/`, Januar 2004.

[Jun94] Dieter Jungnickel. *Graphen, Netzwerke und Algorithmen*. Wissenschaftsverlag, 1994.

[KGH02] Michael Koch, Georg Groh, Christian Hillebrand. Mobile communities - extending online communities into the real world. In *Proceedings of the Americas Conference on Information Systems (AMCIS2002), Dallas, Texas*, August 2002.

[Kob93] A. Kobsa. User modeling: Recent work, prospects and hazards. In M.-Schneider-Hufschmidt, T. Kühme, U. Malinowski, editors, *Adaptive user interfaces: Principles and practice, Netherlands, Amsterdam*, pages 111–128, 1993.

[KR90] L. Kaufman, P. J. Rousseeuw. *Finding Groups in Data: An introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[KVCP97] P. Krishna, N. Vaidya, M. Chatterjee, D. Pradhan. A cluster-based approach for routing in dynamic networks. In *ACM SIGCOMM Computer Communication*, pages 49–65, April 1997.

[LMB+03] Menno Lindwer, Diana Marculescu, Twan Basten, Rainer Zimmerman, Radu Marculescu, Stefan Jung, Eugenio Cantatore. Ambient intelligence visions and achievements: Linking abstract ideas to real-world concepts. In *IEEE Conference and Exhibition on Design, Automation and Test in Europe (DATE'03)*, Munich, Germany, 2003. IEEE Press.

[LS00] Henry Lieberman, Ted Selker. Out of context: Computer systems that adapt to, and learn from context. *IBM Systems Journal*, 39(3&4):617–632, 2000.

[LSG02] Sung Ju Lee, William Su, Mario Gerla. Demand multicast routing protocol in multihop wireless mobile networks. *Mobile Networks and Applications*, 7(6):441–453, Dezember 2002.

[Mai98] R. Maitra. Clustering massive datasets with applications in software metrics and tomography. *American Statistics Association Journal*, 1998.

[Mat89] Friedemann Mattern. *Verteilte Basisalgorithmen*, volume 226 of *Informatik Fachberichte*. Springer-Verlag Berlin, 1989.

[MD96] M. Mari, S. Dellepiane. A segmentation method based on fuzzy topology and clustering. *In Proceedings of the ICPR*, pages 565–569, August 1996.

[Mei91] Christoph Meinel. *Effiziente Algorithmen: Entwurf und Analyse*. Fachbuchverlag Leipzig, 1991.

[MFP04] Tatiana K. Madsen, Frank H. P. Fitzek, Ramjee Prasad. Impact of Different Mobility Models On Connectivity Probability of a Wireless Ad Hoc

Network. In *Proceedings of the International Workshop on Wireless Ad hoc Networks (IWWAN '04)*, Centre of Wireless Communications, University of Oulu, Finland, June 2004.

[Mic00a] Microsoft. *Understanding Universal Plug and Play: A White Paper*, June 2000.

[Mic00b] Microsoft. *Universal Plug and Play Device Architecture, Version 1.0*, June 2000.

[Mic03a] Sun Microsystems. JXME-homepage. `http://jxme.jxta.org`, 2003.

[Mic03b] Sun Microsystems. *Multi-Schema-Validator - MSV*, 2003.

[Mil01] Dejan Milojicic. M. Satyanarayanan on Mobile and Pervasive Computing. *IEEE Distributed Systems Online*, 2(6), 2001.

[Mil02] Dejan Milojicic. A Discussion with Leslie Lamport. *IEEE Distributed Systems Online*, August 2002.

[MKDW03] Alexei A. Makarenko, Tobias Kaupp, Hugh F. Durrant-Whyte. Scalable human-robot interactions in active sensor networks. *IEEE Pervasive Computing*, 2(4):63–71, 2003.

[MM03] Andreas Meissner, Sharath Babu Musunoori. Group integrity management support for mobile ad-hoc communities. In Christina Ururahy, Alexandre, Sztajnberg, Renato Cerqueira, editors, *Workshop proceedings Middleware for Pervasive and Ad-Hoc Computing in conjunction with ACM/IFIP/USENIX International Middleware Conference - Middleware 2003*, pages 53–59, 2003.

[Mot02] Motorola. *Java APIs for Bluetooth Wireless Technology (JSR-82), Version 1.0a*, April 2002.

[MSUB04] Christian Müller-Schloer, Theo Ungerer, Bernhard Bauer, editors. *Towards a General Approach To Mobile Profile Based Distributed Grouping*, volume 2981 of *Lecture Notes in Computer Science*. Springer, 2004.

[MvH04] Deborah L. McGuinness, Frank van Harmelen. *W3C Recommendation - OWL Web Ontology Language*, 2004.

[MWV00] N. Malpani, J. Welch, N. Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the Fourth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 96–103, 2000.

[NH02] Raymond T. Ng, Jiawei Han. CLARANS: a method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, September/Oktober 2002.

[Pal85] Edgar M. Palmer. *Graphical Evolution*. John Wiley & Sons, 1985.

[PB94] Charles E. Perkins, P. Bhagwat. Highly dynamic destination sequenced distance vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94*, 1994.

[PBRD03] Charles E. Perkins, E. Belding-Royer, S. Das. RFC 3561, Ad hoc on-demand distance vector (AODV) routing. `http://www.faqs.org/rfcs/-rfc3561.html`, Juli 2003.

[PBW04] Michael Pirker, Michael Berger, Michael Watzke. An approach for FIPA agent service discovery in mobile ad hoc environments. In *Proceedings of the Conference on Autonomous Agents and Multiagent Systems - Workshop on Agents for Ubiquitous Computing*, New York, USA, 2004.

[PC97] Vincent D. Park, M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of the IEEE INFOCOM*, April 1997.

[Pir03] Michael Pirker. A concept for the interoperability of fipa agents in mobile ad hoc environments. Diplomarbeit, Technische Universität Graz, Institut für Technische Informatik, September 2003.

[PMR99] Roberto De Prisco, Dahlia Malkhi, Michael K. Reiter. On k-set consensus problems in asynchronous systems. In *Symposium on Principles of Distributed Computing*, pages 257–265, 1999.

[Poh96] W. Pohl. Learning about the user – user modeling and machine learning. In V. Moustakis, J. Herrmann, editors, *Proceedings of the Workshop Machine Learning meets Human-Computer Interaction (ICML'96)*, pages 29–40, 1996.

[PR00] Charles E. Perkins, Elizabeth M. Royer. The ad hoc on-demand distance-vector protocol. In Charles E. Perkins, editor, *Ad hoc Networking*, pages 173–219. Addison-Wesley, 2000.

[PSL80] M. Pease, R. Shostak, L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, April 1980.

[RHH01] Gruia-Catalin Roman, Qingfeng Huang, Ali Hazemi. Consistent group membership in ad hoc networks. In *International Conference on Software Engineering*, pages 381–388, 2001.

[RN03] Stuart Russell, Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[RSA78] R. L. Rivest, A. Shamir, L. M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Februar 1978.

[Sat01] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, pages 10–17, August 2001.

[Sat04] M. Satyanarayanan. PERCOM 2004 Keyote-Speech. `http://www.percom.-org/percom_2004/programhighlights.htm#key1`, 2004.

[SAW94] Bill Schilit, Norman Adams, Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.

[SB03] Christian Seitz, Michael Berger. Towards an approach for mobile profile based distributed clustering. In Harald Kosch, László Böszörményi, Hermann Hellwagner, editors, *Euro-Par 2003. Parallel Processing, 9th International Euro-Par Conference, Klagenfurt, Austria, August 26-29, 2003. Proceedings*, volume 2790 of *Lecture Notes in Computer Science*. Springer, 2003.

[SBB04a] Christian Seitz, Michael Berger, Bernhard Bauer. MopiMine - Mobile Profile Mining. In *Proceedings of the International Workshop on Wireless Ad hoc Networks (IWWAN '04)*, Centre of Wireless Communications, University of Oulu, Finland, June 2004.

[SBB04b] Christian Seitz, Michael Berger, Bernhard Bauer. MPDG - Mobile Profile based Distributed Grouping. In *Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communications - Workshop on Mobile Peer-to-Peer Computing*, Orlando, Florida, USA, März 2004. IEEE Computer Society.

[SBB05] Christian Seitz, Michael Berger, Bernhard Bauer. Towards a General Approach To Mobile Profile Based Distributed Grouping. *Personal and Ubiquitous Computing Journal*, to appear 2005.

[SC98] S. J. Soltysiak, I. B. Crabtree. Automatic learning of a user profile - towards the personalisation of agent services. *BT Technological Journal*, 16(3):1998, 1998.

[Seg83] A. Segal. Distributed network protocols. *IEEE Transactions on Information Theory*, IT-29(1):23–35, 1983.

[Sha03] Nigel Shadbolt. Ambient intelligence. *IEEE Intelligent Systems*, pages 2–3, Juli/August 2003.

[SKD00] P. Sinha, S. Krishnamurthy, S. Dao. Scalable unidirectional routing with zone routing protocol. In *In Proceedings of Wireless Communications and Networking Conference (WCNC)*, pages 1329–1339, 2000.

[SM83] G. Salton, M. J. McGill. *Introduction to information retrieval.* McGraw-Hill, Inc., New York, NY., 1983.

[SS94] Mukesh Singhal, Niranjan G. Shivaratri. *Advanced Concepts in Operation Systems.* McGraw-Hill, 1994.

[Sto03] Adam Stone. The dark side of pervasive computing. *IEEE Pervasive Computing - Mobile and Ubiquitous Systems*, 2(1):4–8, 2003.

[Sun02] Sun Microsystems. *Java 2 Platform, Micro Edition*, 2002.

[Sun03a] Sun Microsystems. *Jini Architecture Specification, Version 1.2*, June 2003.

[Sun03b] Sun Microsystems. *JXTA v2.0 Protocols Specification*, March 2003.

[Tan97] A. S. Tanenbaum. *Computernetzwerke*. Prentice Hall, 1997.

[TBMM01] Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn. *W3C Recommendation - XML Schema Part 1: Structures*, 2001.

[Tut84] W. T. Tutte. *Graph Theory*. Addison-Wesley, 1984.

[TY00] Lin Yu Tseng, Shiueng Bien Yang. A genetic clustering algorithm for data with non-spherical-shape clusters. *Pattern Recognition*, 33:1251–1259, 2000.

[vdLPB02] Mark J. van der Laan, Katherine S. Pollard, Jennifer Bryan. A new paritioning around medoids algorithm. Working Paper 105, U. C. Berkely Division of Biostatistic Working Paper Series, 2002.

[VMSF01] Daniel Veit, J. P. Müller, Martin Schneider, B. Fiehn. Matchmaking for autonomous agents in electronic market places. In *Proceedings of the 5th International Conference on Autonomous Agents, Montreal, Canada*, 2001.

[W3C03] W3C. *W3C XML Schema Validator - (XSV)*, 2003.

[WB97] Mark Weiser, John Seeley Brown. The coming age of calm technology. In Peter J. Denning, Robert M. Metcalfe, editors, *Beyond Calculation: The Next Fifty Years of Computing*. Springer Verlag, 1997.

[Wei91] Mark Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–100, September 1991.

[Wei93a] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–85, 1993.

[Wei93b] Mark Weiser. Ubiquitous computing. *IEEE Computer „Hot Topics“*, Oktober 1993.

[Wei94] Mark Weiser. The world is not a desktop. *Interactions*, Januar 1994.

[WIY01] Dwi H. Widyantoro, Thomas R. Ioerger, John Yen. Learning user interest dynamics with a three-descriptor representation. *Journal of the American Society of Information Science*, 52(3):212–225, 2001.

[WQA+02] Y. M. Wang, L. Qiu, D. Achlioptas, G. Das, P. Larson, H. J. Wang. Subscription partitioning and routing in content-based publish/subscribe systems. In *Proceedings of the 16th Symposium on Distributed Computing*, 2002.

[Yoc96] J. Yochum. Research in automatic profile generation and passage-level routing with LMDS, 1996.

Index

Lebenslauf

Persönliche Daten

Name	Christian Georg Seitz
Geburtstag	25. Februar 1975
Geburtsort	Eichstätt
Anschrift	Jurastraße 31 91809 Wellheim
Familienstand	ledig

Schule und Studium

1981 - 1985	Grundschule Wellheim
1985 - 1987	Hauptschule Wellheim
1987 - 1991	Realschule Rebdorf Abschluss: mittlere Reife
1991 - 1993	Fachoberschule Ingolstadt Abschluss: Fachhochschulreife
1993 - 1994	Fachhochschule Rosenheim, Studium der Informatik
November 1995 - Oktober 2001	Friedrich-Alexander-Universität Erlangen-Nürnberg Abschluss: Diplom in Informatik
Oktober 2001 – September 2002	Humboldt Universität zu Berlin Angestrebter Abschluss: Promotion Informatik
seit Oktober 2002	Universität Augsburg Angestrebter Abschluss: Promotion Informatik

Arbeitserfahrung

August 1997 - September 1997	Werkstudent im Bereich Quality of Service Überwachung, AUDI AG, Ingolstadt
September 2000 - Februar 2000	Diplomarbeit im Fachzentrum für ‚Intelligente Autonome Systeme', Siemens AG, Corporate Technology, München
März 2001 - April 2001	Werkstudent im Fachzentrum für ‚Intelligente Autonome Systeme', Siemens AG, Corporate Technology, München
seit Mai 2001	Promotion im Fachzentrum für ‚Intelligente Autonome Systeme', Siemens AG, Corporate Technology, München Thema: „Ein Framework für die profilbasierte Gruppenbildung in ad hoc Umgebungen"

Sonstiges

Grundwehrdienst	Oktober 1994 - September 1995 in Neuburg/Donau
Sprachen	Englisch

München, im April 2005

Christian Seitz